THE IMPLICATION PROBLEM FOR DATA DEPENDENCIES

Extended Abstract

C. Beeri and M.Y. Vardi*

Department of Computer Science
The Hebrew University of Jerusalem
Jerusalem 91904, Israel

ABSTRACT

In this paper we study the implication and the finite implication problems for data dependencies. When all dependencies are total the problems are equivalent and solvable but are NP-hard, i.e., probably computationally intractable. For non-total dependencies the implication problem is unsolvable, and the finite implication problem is not even partially solvable. Thus, there can be no formal system for finite implication. The meta decision problems of deciding for a given class of dependencies whether the implication problem is solvable or whether implication is equivalent to finite implication are also unsolvable.

1. INTRODUCTION

One of the important issues in the design of relational database schemas is the specification of the constraints that the data must satisfy to model correctly the part of the world under consideration. These constraints determine which databases are considered meaningful.

Of particular interest are the constraints called data dependencies. The first class of dependencies to be studied was the class of functional dependencies [Codd], which was followed by the class of multivalued dependencies [Fag1,Zan]. Recently, a number of generalizations of these dependencies have appeared; e.g., join dependencies [ABU,Riss], general dependencies [JP], and template dependencies [SU]. All these classes are subclasses of the class of tuple and equality generating dependencies of [BV2,Fag2,YP]. Intuitively, the meaning of a dependency is that if some tuples, fulfilling certain conditions, exist in the database, then either some other tuples must also exist therein, or some values in the given tuples must be equal.

A utilization of the above dependencies in the design of a relational database requires algorithms for determining whether a set of dependencies is redundant

*Research partially supported by Grant 1849/79 of the U.S.A.-Israel Binational Science Foundation.

From Proc. ICALP 1981, Acre, Israel
in Lecture Note in CS 115, pp. 73-85

([Ber]) and whether two sets of dependencies are equivalent ([BMSU,BeRi]). Both problems reduce to the implication problem, i.e., the problem of deciding whether a given set of dependencies logically implies another dependency. The finite implication problem is the problem of implication when only finite relations are taken into account.

The formalism is that of first order logic. We do not show how the various dependencies mentioned above can be written in this formalism, and the reader interested in that aspect is refered to [BV2,Nico]. In fact, we mostly refrain from using "relational" terminology, and except for a few remarks this paper is essentially concerned with a fragment of first order logic, which is relevant to database theory.

This paper is an abridged version of [BV1].

2. DEPENDENCIES

We use the language $L(n)$ of first order logic with equality with no function symbols and one n-ary predicate symbol. Indexed x's are used for existentially quantified variable symbols, and indexed y's are used for universally quantified variable symbols. Indexed v's are syntactical variables ranging over variable symbols. An atomic formula $R(v_1,\ldots,v_n)$ is called a predicate formula and an atomic formula $v_i=v_j$ is called an equality formula. A dependency is a sentence

$\forall y_1\ldots\forall y_k\ \exists x_1\ldots\exists x_\ell(A_1\wedge\ldots\wedge A_p \rightarrow B_1\wedge\ldots\wedge B_q)$, where:

(a) $k,p,q \geq 1, \ell \geq 0$.

(b) the A's and the B's are atomic formulas.

(c) at least one A_i is a predicate formula.

(d) the set of variables occuring in the A's is the same as the set of variables occuring in the predicated A's, and is exactly $\{y_1,\ldots,y_k\}$.

(e) the set of variables occuring in the B's contains $\{x_1,\ldots,x_\ell\}$.

Restrictions (c) and (d) ensure that the sentence refers only to the information contained within the database.

Suppose now that some A_r is $y_i=y_j$. Obviously, we can identify y_i and y_j wherever they occur in the dependency, and eliminate A_r to get an equivalent dependency. Thus, we can assume

(f) all the A's are predicate formulas.

Suppose now that some B_r is $x_i=v_j$. Again, we can identify x_i and v_j and eliminate B_r to get an equivalent dependency. Thus, we can assume:

(e) all equality formulas are of the form $y_i = y_j$.

Finally, recalling that $\forall y(A \to B \wedge C)$ is equivalent to $\forall y(A \to B) \wedge \forall y(A \to C)$, if y is free in A, B and C, we assume:

(f) either all the B's are predicate formulas or $q=1$ and B_1 is an equality formula.

Intuitively, the meaning of a dependency is that if some tuples, fulfilling certain conditions, exist in the database, then either some other tuples must also exist therein, or some values in the given tuples must be equal.

We now distinguish between several subclasses of dependencies. This is summarized in the following table.

case	name	abbr
all B's are predicate formulas	tuple generating	tgd
$q=1$ and B_1 is an equality formula	equality generating	egd
$\ell=0$ (no existential quantifier)	total	td
$q=1$	many to one	mod
$p=2$ and $q=1$	two to one	tod
many sorted (see definition)	many sorted	msd

A dependency is <u>many sorted</u> if no variable occurs in two different argument positions of the predicate symbol, and only variables which occur in the same argument position of the predicate symbol can be the arguments of an equality formula. Almost all dependencies dealt with in the literature are msd's. For example, for msd's:

(1) an egd with $p=2$ is a functional dependency [Codd].

(2) a tgd with $q=1$ is a template dependency [SU].

<u>Remark</u>. One may ask whether our syntactic definitions for dependencies can be replaced by semantic definitions. To a certain degree this can be done [CLM]. However, semantic definitions can characterize only up to logical equivalence, and the set of first order sentences equivalent to some dependency is not recursive. <>

The dependencies of $L(n)$ are called n-ary dependencies. In studying decision problems for $L(n)$, n may be either a parameter of the problem or some fixed value. The class of all dependencies is denoted Dep. In the sequel we use D to denote a finite set of dependencies and d, d' to denote single dependencies. In writing down dependencies we usually omit universal quantifiers.

3. IMPLICATION PROBLEMS

Let $U = \langle A,R \rangle$ be a structure for $L(n)$. U finite if A is finite (and consequently, R is finite). U is semifinite if R is finite (A can be infinite). U is infinite if R is infinite (and obviously, A is infinite). U is empty if R is empty, and is trivial if it is empty or if $|A| = 1$. (Note that A is always assumed to be nonempty).

A set of dependencies D implies a dependency d, denoted $D \models d$, if d holds in all models of D. D semifinitely implies d, denoted $D \models_{sf} d$, if d holds in all semifinite models of D.

D finitely implies d, denoted $D \models_{f} d$, if d holds in all finite models of D. Clearly, real-life databases are finite, but the domain of values might be conceptually infinite. However, for dependencies $\models_{sf}$ and $\models_{f}$ are equivalent.

Lemma 1. $D \models_{sf} d$ iff $D \models_{f} d$. <>

By Lemma 1 it suffices to deal with $\models$ and $\models_{f}$. Our decision problems are:

(a) The implication problem - for a given D and d, decide whether $D \models d$.

(b) The finite implication problem - for a given D and d, decide whether $D \models_{f} d$.

The (finite) implication problem of type $(C_1 ; C_2)$, where C_1 and C_2 are classes of dependencies is the (finite) implication problem for $D \subseteq C_1$ and $d \in C_2$. That is, for such D,d decide whether $D \models_{(f)} d$.

As is well-known, both the implication and the finite implication problems are unsolvable for arbitrary first order sentences. Note that $D \models d$ entails $D \models_{f} d$, but not vice versa, hence, the implication and the finite implication problems are independent. In fact, their equivalence entails their solvability.

Lemma 2. The following sets are recursively enumerable:

(a) $\{\langle D,d \rangle \mid D \models d\}$.

(b) $\{\langle D,d \rangle \mid D \not\models_{f} d\}$. <>

Corollary. If for classes of dependencies C_1 and C_2 we have that for $D \subseteq C_1$ and $d \in C_2$, $D \models d$ iff $D \models_{f} d$, then the implication problem of type $(C_1 ; C_2)$ is equivalent to the finite implication problem and is solvable. <>

Let us now consider the case where D is the empty set. A dependency d is trivial if it holds in all structures, denoted $\models d$, and is finitely trivial if it holds in all finite structures, denoted $\models_f d$. Thus, as special cases of the (finite) implication problem, we get:

(a) The <u>triviality problem</u> - for a given d, decide whether d is trivial.

(b) The <u>finite triviality problem</u> - for a given d, decide whether d is finitely trivial.

4. SOME SOLVABLE CASES

If we restrict D to be a set of td's, then the (finite) implication problem is equivalent to the (finite) validity problem for $\exists^*\forall^*$ sentences (Schonfinkel-Bernays class), whose solvability follows from Lemma 2 [BS].

<u>Theorem 1</u>. The implication problem of type (td's ; Dep) is equivalent to the finite implication problem, and is solvable. <>

As a special case we get the solvability of the (finite) triviality problem.

<u>Theorem 2.</u> A dependency d is trivial iff it is finitely trivial iff

(a) d is a egd and B_1 is $y_i=y_i$, or

(b) d is a tgd and for some substitution sequence $1 \leq i_1,\ldots,i_\ell \leq k$, $\{B_1,\ldots,B_q\}\ (x_1/y_{i_1},\ldots,x_1/y_{i_1}) \subseteq \{A_1,\ldots,A_p\}$. <>

A decision procedure for the implication problem of type (td's ; Dep) is described in [BV2]. In some more restricted cases there is an efficient decision procedure [BB,Beer,BV2,MSY,Va], but this is not the case in general. We provide now some upper and lower time bounds.

The following upper bound follows from the complexity analysis of the above mentioned decision procedure [BV2].

<u>Theorem 3</u>. Let D be a set of n-ary td's with u universal quantifiers, and let d be an n-ary dependency with p universal quantifiers and e existential quantifiers. Let s be the number of symbols in D and d. The implication problem for D and d can be solved in time $O(s\cdot p^{2n+u+e})$. <>

The following theorems imply that, except in some restricted cases, there is probably no efficient decision procedure for the implication problem for this solvable case.

<u>Theorem 4</u>. The triviality problem for tgd's is NP-complete, even for msd's and binary dependencies.

<u>Proof</u>: In NP: Nondeterministically choose a substitution sequence and check for the condition of the Theorem 2.

Hard for NP:

(a) msd's: reduction from EXACT COVER [Ka].

(b) binary dependencies: reduction from CLIQUE [Ka]. <>

Theorem 5. The set $\{\langle d,d'\rangle \mid d,d'$ are total mod's and $d \models d'\}$ is NP-hard even for msd's and binary dependencies.

Proof: We use the following NP-complete problems for reduction:

(a) Msd's: reduction from EXACT COVER [Ka].

(b) Binary dependencies: reduction from CLIQUE [Ka]. <>

Additional results on the complexity of testing implication of msd's can be found in [BV3].

In some cases solvability follows from the fact that the answer to the decision problem is trivially negative:

Lemma 3. Let D be a set of tgd's, and let d be an egd, then $D \models d$ if and only if $D \models_f d$ if and only if d is trivial. <>
For several other solvable cases see [BV2].

When dealing with implication of tgd's, we can very easily eliminate egd's from consideration. Let d be the egd $\forall y_1 \ldots \forall y_k (A_1 \wedge \ldots \wedge A_p \rightarrow y_g = y_h)$. Let A denote the predicate formula $R(y_{k+1},\ldots,y_{k+n})$, and denote by $A(m/y_i)$, for $1 \leq m \leq n$, the result of substituting y_i for y_{k+m} in A. We associate with d the following set of tgd's: D_1 is $\{\forall y_1 \ldots \forall y_{k+n} (A_1 \wedge \ldots \wedge A_p \wedge A(m/y_g) \rightarrow A(m/y_h)) \mid 1 \leq m \leq n\}$, D_2 is defined similarly, with g and h interchanged, and D_d is taken to be the union of D_1 and D_2. Let D be a set of dependencies, we denote by D^* the result of replacing each egd d in the set D by D_d.

Lemma 4. Let D be a set of dependencies and d a tgd, then $D \models d$ iff $D^* \models d$ and $D \models_f d$ iff $D^* \models_f d$. <>

It is well known that equality can be eliminated from first-order logic by adding the equality axioms: reflexivity, symmetry, transitivity and substitutivity. This can also be applied to dependencies. Actually, we can prove an even stronger result:

Theorem 6. Let D be a set of dependencies, and let d be a dependency. We can effectively construct a set of tuple generating tod's D' and a tuple generating tod d', such that $D \models d$ if and only if $D' \models d'$, and $D \models_f d$ if and only if $D' \models_f d'$. <>

5. UNSOLVABILITY RESULTS

The main result of this section is:

Theorem 7. The implication and the finite implication problems are unsolvable. <>

Unsolvability is shown by encoding appropriate unsolvable problems of equational logic in terms of depe encies.

Let L_{eq} be the language of first order logic with equality, with function symbols but no individual constants or predicate symbols. An equation is a sentence $\forall y_1 \ldots \forall y_k (s=t)$, where s and t are terms of L_{eq}. A conditional equation is a sentence $\forall y_1 \ldots \forall y_k (s_1=t_1 \wedge \ldots \wedge s_{m-1} = t_{m-1} \rightarrow s_m = t_m)$, $m > 1$, where $s_1, t_1, \ldots, s_m, t_m$ are terms of L_{eq}. Equational logic is a fragment of first order logic, in which equations and conditional equations are the only admitted sentences.

Let L_2 be L_{eq} with one binary function symbol g. A conditional equation of L_2 is simple if it is of the form $\forall y_1 \ldots \forall y_k (e(1) \wedge \ldots \wedge e(m-1) \rightarrow e(m))$, $m > 1$, where $e(i)$ is $g(v_i^1, v_i^2) = v_i^3$ for $1 \leq i < m$, and $e(m)$ is $v_k^p = v_\ell^q$, $1 \leq k, \ell < m$, $1 \leq p, q \leq 3$.

Lemma 5. For every (conditional) equation of L_2 we can effectively construct an equivalent simple conditional equation. <>

A structure $U = \langle A, f_1, f_2, \ldots \rangle$ for L_{eq} is finite if A is finite, and is trivial if $|A| = 1$. Clearly, every (conditional) equation has a trivial model. Non-trivial consistency is, however, unsolvable.

Theorem 8. [McKe] The following two problems are unsolvable for L_2:

(a) to decide if an equation has a non-trivial model.

(b) to decide if an equation has a non-trivial finite model. <>

Corollary. The above problems are unsolvable even for simple conditional equations. <>

Equations can be coded by dependencies by replacing functions by their representing relations. Let $U = \langle A, g \rangle$ be a structure for L_2, i.e., U is a groupoid. The representing relation for U is a ternary relation

$$G = \{\langle x,y,z \rangle \mid z = g(x,y)\} .$$

G satisfies the following condition:

(*) For all x, y, each belonging to some triple in G, there exists a unique z such that $\langle x,y,z \rangle \in G$.

Conversely, any non-empty ternary relation G on a set B satisfying (*) defines a groupoid $U = \langle A, g \rangle$, where $A = \{x \mid \langle x,y,z \rangle \in G\} \subseteq B$, and $g(x,y) = z$, where z is the unique element such that $\langle x,y,z \rangle \in G$.

Condition (*) is expressed by the following dependencies:

G1: $\exists x(G(y_1,y_2,y_3) \rightarrow G(y_2,y_3,x))$

G2: $\exists x(G(y_1,y_2,y_3) \wedge G(y_4,y_5,y_6) \rightarrow G(y_5,y_1,x))$

G3: $G(y_1,y_2,y_3) \wedge G(y_1,y_2,y_4) \rightarrow y_3=y_4$

Let Eq: $\forall y_1 \ldots \forall y_k(e(1) \wedge \ldots \wedge e(m-1) \rightarrow e(m))$ be a simple conditional equation. To express it in terms of the representing relation we replace the equality formula $e(i)$ by the predicate formula $E(i)$: $G(v_i^1,v_i^2,v_i^3)$ to get the <u>representing dependency</u> d_{Eq}: $\forall y_1 \ldots \forall y_k\ (E(1) \wedge \ldots \wedge E(m-1) \rightarrow e(m))$.

<u>Lemma 6</u>. Let $U = \langle A,g \rangle$ be a non-trivial (finite) groupoid satisfying a simple conditional equation Eq, then its representing relation G satisfies $\{G1,G2,G3,d_{Eq}\}$. Conversely, if G is a non-trivial (finite) ternary relation satisfying $\{G1,G2,G3,d_{Eq}\}$, then it defines a non-trivial (finite) groupoid satisfying Eq. <>

As an immediate consequence we get:

<u>Theorem 9</u>. The following two problems are unsolvable even for ternary mod's:

(a) to decide if a set of dependencies D has a non-trivial model.

(b) to decide if a set of dependencies D has a non-trivial finite model. <>

This result will serve as a springboard for proving the unsolvability of the implication and the finite implication problems. However, it does have a significance by itself, since if a database is described by a set of dependencies which have no (finite) non-trivial model, then this set is probably semantically meaningless.

Let Ga be {G1,G2,G3}, and let Gb be Ga^* (i.e., Gb is the result of replacing G3 by tgd's as described in Section 4). We define two dependencies:

T1: $G(y_1,y_2,y_3) \rightarrow y_1 = y_2$,

T2: $G(y_1,y_2,y_3) \wedge G(y_1,y_4,y_5) \rightarrow G(y_1,y_2,y_4))$.

<u>Theorem 10</u>. The following sets of ternary tuple generating mod's are not recursive:

(a) $\{d \mid Ga \cup \{d\} \models T1\}$,

(b) $\{d \mid Ga \cup \{d\} \models_f T1\}$,

(c) $\{d \mid Gb \cup \{d\} \models T2\}$,

(d) $\{d \mid Gb \cup \{d\} \models_f T2\}$,

<u>Proof</u>. A groupoid is trival iff it satisfies the equation $\forall x \forall y(x = y)$ iff it satisfies the equation $\forall x \forall y \forall z(g(x,y) = z)$. Since T1 and T2 represent these equations, the claim follows by Theorem 8 and Lemma 6. <>

The meaning of the above theorem is that the set of dependencies implying a specific dependency is not recursive. We are going now to construct a set of dependencies Gc, such that the set of dependencies implied by Gc is not recursive.

A <u>group</u> is a groupoid satisfying the following axioms [TMR]:

H1: $g(x,g(y,z)) = g(g(x,y),z$

H2: $\exists z(x = g(y,z))$,

H3: $\exists z(x = g(z,y))$.

These axioms are expressed by the following dependencies:

G4: $G(y_2,y_3,y_4) \wedge G(y_1,y_4,y_5) \wedge G(y_1,y_2,y_6) \rightarrow G(y_6,y_3,y_5))$,

G5: $\exists x(G(y_1,y_2,y_3) \rightarrow G(y_2,x,y_1))$,

G6: $\exists x(G(y_1,y_2,y_3) \rightarrow G(x,y_2,y_1))$.

The following theorem is the well-known unsolvability result for the word problem for groups (e.g. [Bo]).

<u>Theorem 11</u>. The set of conditional equations which holds in all groups in not recursive. <>

Let Gc be {G1,...,G6} . Using Lemma 5 we get:

<u>Theorem 12</u>. The following set of ternary egd's is not recursive:
$\{d \mid Gc \models d\}$. <>

6. MORE UNSOLVABILITY RESULTS

Actually, we have proved in the previous section a result which is stronger than Theorem 7.

<u>Theorem 13</u>. The (finite) implication problem for ternary tuple generating mod's is unsolvable. <>

By using various reduction technique, we can also have:

<u>Theorem 14</u>. The (finite) implication problem for binary tgd's and for 5-ary tuple generating msd's is unsolvable. <>

<u>Remark</u>. When constants are allowed to appear in dependencies (two constants suffice), the (finite) implication problem is unsolvable even for 4-ary many-sorted tuple generating mod's. <>

For some sets of dependencies D_1,D_2, the set

$$IMPL(D_1,D_2) = \{d \mid d \in D_2 \text{ and } D_1 \models d\}$$

may be recursive. The <u>meta implication problem</u> is to decide, for given recursive sets of dependencies D_1,D_2, whether $IMPL(D_1,D_2)$ is recursive.

<u>Theorem 15</u>. The meta implication problem is unsolvable.

<u>Proof</u>. The claim follows from the unsolvability of the meta word problem for groups [Ra]. <>

Combining our unsolvability results with Lemma 2 we get:

<u>Theorem 16</u>. The following sets are not recursively enumerable:

(a) $\{\langle D,d\rangle \mid D \models_f d\}$
(b) $\{\langle D,d\rangle \mid D \not\models d\}$. <>

From part (a) of the theorem it follows that there is no proof procedure for finite implication of dependencies, and obviously no sound and complete formal system for finite implication can be found. In contrast, a proof procedure and a formal system for implication does exist [BV2,BV4,YP].

By the corollary of Lemma 2, $\models$ and $\models_f$ are not equivalent for dependencies in general, and by Theorem 1 they are equivalent for some classes of dependencies. The <u>implication equivalance problem</u> is to decide, for given recursive sets of dependencies D_1,D_2, whether for all $d \in D_2$, $D_1 \models d$ iff $D_1 \models_f d$.

<u>Theorem 17</u>. The implication equivalence problem is unsolvable.

<u>Proof</u>: The claim follows from the unsolvability of the residual finiteness problem for groups. [Ra]. <>

We conclude by showing that $\models$ and $\models_f$ are not equivalent even for binary mod's, though the solvability issue for this class is open. We use d_1,d_2,d_3,d_4 and d_5:

d_1: $\exists x(R(y_1,y_2) \to R(y_2,x))$,

d_2: $R(y_1,y_2) \wedge R(y_2,y_3) \to R(y_1,y_3)$,

d_3: $R(y_1,y_1) \wedge R(y_2,y_3) \to R(y_3,y_2)$,

d_4: $\exists x(R(y_1,y_2) \to R(x,x))$

d_5: $R(y_1,y_2) \to R(y_2,y_1)$.

<u>Lemma 10</u>.

(a) $\{d_1,d_2\} \models_f d_4$ but $\{d_1,d_2\} \not\models d_4$,
(b) $\{d_1,d_2,d_3\} \models_f d_5$ but $\{d_1,d_2,d_3\} \not\models d_5$. <>

7. CONCLUDING REMARKS

The originators of dependency theory intended to develop a tool for automated database design. Our lower bounds for (finite) implication indicate that in its present state the theory is far from being such a tool. Thus, the theory has not yet passed its "true test" which is "demonstrating its effectiveness in solving day to day database design problems" [BBG].

It should be noted however, that while unsolvability holds for fairly restricted classes of dependencies, we could not extend it for many-sorted mod's, and, more specifically, to embedded multivalued dependencies ([Fag1]) and embedded join dependencies ([MMS]). It is known that the (finite) implication problem for embedded multivalued and join dependencies of any fixed arity is solvable. It is also known ([YP]) that unsolvability for the class of many-sorted mod's entails unsolvability for a class which is slightly more general than the class of embedded join dependencies.

The implication problem is a "local" decision problem. As said in the introduction, our motivation for studying it was the search for algorithms to solve "global" decision problems, the equivalence problem and the redundancy problem. Since $D \models d$ iff $D \cup \{d\} \models\!\!\dashv D$, unsolvability of the implication problem entails unsolvability of the equivalence problem. This is not the case for the redundancy problem. That and other "global" decision problems will be dealt with in a future paper.

Acknowledgements.

We are grateful to J. Makowsky for fruitful discussions and to D. Harel for helpful comments.

REFERENCES

[ABU] Aho, A.V., Beeri, C., Ulman, J.D.: The theory of joins in relational databases. ACM TODS 4(1979), pp. 297-314.

[BB] Beeri, C., Bernstein, P.A.: Computational problems related to the design of normal form relational schemas. ACM TODS 4(1979), pp. 30-59.

[BBG] Beeri, C., Bernstein, P.A., Goodman, N.: A sophisticate's introduction to database normalization theory. Proc. 4th Conf. on VLDB, 1978, pp.113-124.

[Beer] Beeri, C.: On the membership problem for multivalued dependencies. ACM TODS 5(1980), pp. 241-259.

[Ber] Bernstein, P.A.: Synthesizing third normal form relation from functional dependencies. ACM TODS 1(1976), pp. 277-298.

[BeRi] Beeri, C., Rissanen, J.: Faithful representation of relational database schemes. IBM Research Report, San Jose, 1979.

[BMSU] Beeri, C., Mendelzon, A.O., Sagiv, Y., Ullman, J.D.: Equivalence of relational database schemes. Proc. 11th ACM STOC, 1979, pp. 319-329.

[Bo] Boone, W.W.: The word problem. Ann. of Math. 70(1959), pp. 207-265.

[BS] Bernays, P., Schonfinkel, M.: Zum Eintscheidungsproblem der Mathematischen Logik. Mat. Annal. 99(1928), pp. 342-372.

[BV1] Beeri, C., Vardi, M.Y.: The implication problem for data dependencies. Proc. XP1 Workshop, 1980. Also, Research Report, The Hebrew University of Jerusalem, 1980.

[BV2] Beeri, C., Vardi, M.Y.: A proof procedure for data dependencies. Research Report, The Hebrew University of Jerusalem, 1980.

[BV3] Beeri, C., Vardi, M.Y.: On the complexity of testing implication of data dependencies. Research Report. The Hebrew University of Jerusalem, 1980.

[BV4] Beeri, C., Vardi, M.Y.: Axiomatization of tuple and equality generating dependencies. Research Report, The Hebrew University of Jerusalem, 1981.

[CLM] Chandra, A.K., Lewis, H.R., Makowsky, J.A.: Embedded implicational dependencies and their inference problem. Proc. XP1 Workshop, 1980. Revised, Proc. 13th ACM STOC, 1981.

[Codd] Codd, E.F.: Further normalization of the data base relational model. in Data Base Systems (R. Rustin, ed.), Prentice-Hall, N.J., 1972, pp. 33-64.

[Fag1] Fagin, R.: Multivalued dependencies and a new normal form for relational databases. ACM TODS 2(1977), pp. 262-278.

[Fag2] Fagin, R.: Horn clauses and database dependencies. Proc. 12th ACM STOC, 1980, pp. 123-134.

[JP] Janssens, D., Paredaens, J.: General dependencies. Workshop on Formal Bases for Databases, Toulouse, Dec. 1979.

[Ka] Karp, R.M.: Reducibility among combinatorial problems. in Complexity of Computer Computation (R.E. Miller and J.W. Thatcher, eds.) Plenum Press, 1972, pp. 85-103.

[McKe] McKenzie, R.: On spectra, and the negative solution for identities having a finite non-trivial model. J. of Symbolic Logic 40(1975), pp. 186-196.

[MMS] Maier, D., Mendelzon, A.O., Sagiv, Y.: Testing Implications of data dependencies. ACM TODS 4(1979), pp. 455-469.

[MSY] Maier, D., Sagiv, Y., Yannakakis, M.: On the complexity of testing implications of functional and join dependencies. to appear in JACM.

[Nico] Nicolas, J.M.: First order logic formalization for functional, multivalued and mutual dependencies. Proc. ACM-SIGMOD, 1978, pp. 40-46.

[Ra] Rabin, M.O.: Recursive unsolvability of grc[illegible]-theoretic problems. Ann. of Math. 67(1958), pp. 172-194.

[Riss] Rissanen, J.: Theory of relations for databases - a tutorial survey. Proc. 7th Symp. on MFCS, Poland, 1978, Lecture Notes in Computer Science 64, Springer-Verlag, pp. 537-551.

[SU] Sadri, P., Ullman, J.D.: A complete axiomatization for a large class of dependencies in relational databases. Proc. 12th ACM STOC, 1980, pp.117-122.

[TMR] Tarski, A., Mostowski, A., Robinson, R.M.: Undecidable theories. North-Holland, Amsterdam, 1953.

[Va] Vardi, M.Y. Inferring multivalued dependencies from functional and join dependencies. Research Report, The Weizmann Institute of Science, 1980.

[YP] Yannakakis, M., Papadimitriou, C.: Algebraic dependencies. Proc. 21st IEEE Symp. on FOCS, 1980, pp. 328-332.

[Zan] Zaniolo, C.: Analysis and design of relational schemata for database systems. Technical Report UCLA-ENG-7769, UCLA, 1976.

Related Work

The unsolvability of the (finite) implication problem for 6-ary mod's and for msd's has been proven independently by Chandra et al. [CLM] by reduction from the halting problem for two-counter machines. They have also shown that the implication problem for total tuple generating msd's is logspace complete in EXPTIME. See also Makowsky's paper in this volume.