THE IMPLICATION PROBLEM FOR DATA DEPENDENCIES

Preliminary Report

by

C. Beeri and M.Y. Vardi*

Department of Computer Science

The Hebrew University of Jerusalem

Jerusalem, Israel

May 1980

---

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

ABSTRACT

In this paper we study the implication problem for data dependencies. We show that the problem is unsolvable even for a fairly restricted class of dependencies. For some subclasses of dependencies the problem is shown to be solvable but probably computationally intractable. Deciding whether a dependency is trivial is NP-complete, and deciding whether a dependency is implied by a total dependency is NP-hard. We prove that two meta decision problems related to implication are unsolvable. Finally, we show that a subclass of dependencies of a very simple structure is a reduction class for the implication problem.

paper. We show that it is unsolvable by reducing to it unsolvable problems of equational logic. Actually, some sets of dependencies characterized by implication are shown to be non-recursive. We also show several solvable subclasses, and provide some complexity bounds.

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 [Nico2]. 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.

The outline of the paper is as follows. In Section 2 we define dependencies, and distinguish between several classes of dependencies. In Section 3 we introduce our decision problems: the implication, finite implication, triviality and finite triviality problems. Some solvable cases are shown in Section 4, however, it is shown that they are probably computationally intractable. The main result of the paper - the unsolvability of the implication and the finite implication problems is proven in Section 5, and in Section 6 we sharpen this result. The inequivalence of implication and finite implication is demonstrated in Section 7. In Section 8 we describe a reduction class for (finite) implication. Finally, in Section 9 we pose some open problems.

In this preliminary report most of the proof are either omitted or briefly sketched.

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 dependencies to be studied were the *functional dependencies* [Codd], which were followed by the *multivalued dependencies* [Fag,Zan]. Recently, a number of generalizations of these dependencies have appeared: *mutual dependencies* [Nico1], *join dependencies* [ABU,Riss], *transitive dependencies* [Pa], *general dependencies* [JP], *subset dependencies* [SW], and *template dependencies* [SU]. In this paper we define *tuple generating dependencies* and *equality generating dependencies* which generalizes all the above mentioned dependencies. 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 in the database, or some values in the given tuples must be equal. The dependency can be either interrelational or intrarelational.

A utilization of the above dependencies in the design of a relational database requires an algorithm for solving the *implication problem*, i.e.,to decide whether a given set of dependencies logically implies another dependency. In some cases this problem is known to be solvable ([MMS,BV]). It is the general case with which we deal in this

2. DEPENDENCIES

We use the language L of first order logic with identity with no function symbols. The signature of the language is the sequence of arities of its predicate symbols, i.e., a language with signature <1,2> has one unary and one binary predicate symbols, denoted L(1,2). 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 $R(v_1,\ldots,v_m)$ is called a predicated formula and an atomic formula $v_i=v_j$ is called an identity formula. A dependency is a sentence

$\forall y_1 \ldots \forall y_k \exists x_1 \ldots \exists x_l (A_1 /\backslash \ldots /\backslash A_p \rightarrow B_1 /\backslash \ldots /\backslash B_q)$, where:

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

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

c) at least one A_i is a predicated 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 $\{y_1,\ldots,y_k\}$.

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

e) all the A's are predicated 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:

f) all identity formulas are of the form $y_i=y_j$.

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

g) either all the B's are predicated formulas or q=1 and B_1 is an identity 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 in the database, or some values in the given tuples must be equal. The dependency can be either interrelational or intrarelational.

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

case	name	abbr.
all B's are predicated formulas	tuple generating	tgd
q=1 and B_1 is an identity formula	equality generating	egd
l=0 (no existential quantifier)	total	td
l > 0	partial	pd
q=1	many to one	mod
p=1	one to many	omd
p=1 and q=1	one to one	ood
each variable y_i, $1 \leq i \leq k$, has a unique occurence in the A's	cross	cd
cross dependency with q=1	cross to one	cod
many sorted (see definition)	many sorted	msd

Let D be a set of dependencies with $\underline{R}=\{R_1,\ldots,R_m\}$ the set of predicate symbols occuring in D. For each $R \in \underline{R}$, let n_R be the arity of R. Consider the set of argument positions $Pos=\{\langle R,i\rangle \mid R \in \underline{R},\ 1 \leq i \leq n_R\}$. Let $\equiv$ be a binary relation defined on Pos in the following way. $\langle R_i,f\rangle \equiv \langle R_j,g\rangle$ if for some $d \in D$ there are variables v_1,v_2 such that v_1 occurs as the f-th argument of R_i, and v_2 occurs as the g-th argument of R_j, and either v_1 and v_2 are identical or $v_1=v_2$ occurs in d. We extend $\equiv$ to its transitive closure. D (and every $d \in D$) is <u>many sorted</u> if for no $R \in \underline{R}$ and $1 \leq i,j \leq n_R$, $i \neq j$, we have $\langle R,i\rangle \equiv \langle R,j\rangle$. Intuitively, that means that the set of variables can be partitioned to different sorts, and all arguments of each predicate symbol must be of different sorts. Observe that this property depends on the collection of dependencies under consideration and not on a single

dependency.

From now on, unless explicitly stated otherwise, we consider only languages L(n) with one n-ary predicate symbol, say R. The arity of a dependency is the arity of its predicate symbol, i.e., a binary dependency is a dependency with a binary predicate symbol. Observe that a dependency with a single predicate symbol is an msd if no variable occurs in two different argument positions, and only variables which occur in the same argument position can be the arguments of an identity.

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 p=2 and q=1 is a subset dependency [SW].

In section 6 we will study the consequences of enriching the language with individual constants. Specifically, we will allow constants to be arguments of a predicate symbol.Unless explicitly specified otherwise, no constants are used.

In the sequel we use d to denote a dependency, and D to denote a set of dependencies.

3. IMPLICATION PROBLEMS

Let $U=\langle A,R_1,\ldots,R_m\rangle$ be a structure for a language with predicate symbols $R_1,\ldots,R_m$ (the distinction between a predicate symbol and its interpretation is left to the reader). U is <u>finite</u> if A is finite (and consequently, all R's are finite). U is <u>semifinite</u> if all the R's are finite (A can be infinite). U is <u>infinite</u> if at least some R_i is infinite (and obviously, A is infinite). U is <u>empty</u> if all the R's are empty, and is <u>trivial</u> if it is empty or if $|A|=1$. (Note that A is always assumed to be nonempty).

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

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

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

a) The <u>implication problem</u> - for a given D and d, decide whether $D \models d$.

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

The (finite) implication problem of <u>type</u> $(C_1 ; C_2)$, where C_1 and C_2 are classes of dependencies is the (finite) implication problem

for $D \subseteq C_1$ and $d \in C_2$. The class of all dependencies is denoted by Dep.

As is known, both the implication and the finite implication problems are unsolvable for arbitray first order sentences [Ch,Tr]. 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) $\{\langle D,d\rangle \mid D \not\models_f d.\ \langle\rangle$

Corolarry. 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. $\langle\rangle$

Let us now consider the case that 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 triviality problem - for a given d, decide whether d is trivial.

b) The finite triviality problem - 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 $\forall^* \exists^*$ sentences (Schonfinkel-Bernays class), whose solvability follows from Lemma 2 [BS].

Theorem 1. 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.

Theorem 2. 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 $1 \leq i_1,\ldots,i_l \leq k$, $\{B_1,\ldots,B_q\}(x_1/y_{i_1},\ldots,x_l/y_{i_l}) \subseteq \{A_1,\ldots,A_p\}$. <>

Remark. The (finite) implication problem of type (td's ; Dep) is solvable also for a language with an arbitrary signature.

A decision procedure for the implication problem of type (td's ; Dep) is decribed in [BV,MMS,SU]. In some more restricted cases there is an efficient decision procedure [BB,Beer,MSY,Va], but this is not the case in general. We provide now some upper and lower time bounds.(1)

---

(1) The complexity of the satisfiability problem for the Schonfinkel-Bernays class is investigated in [Le,Pl].

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

Theorem 3. Let D be a set of td's and let d be a dependency d, where the arity of the dependencies is n, d has p universal quantifiers and e existential quantifiers, D has u universal quantifiers, and the number of symbols in D and d is s; then the implication probelm for D and d can be solved in $O(s \cdot p^{2n+u+e})$ time. <>

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

Theorem 4. The triviality problem for tgd's is NP-complete, even in the following restricted cases:

a) msd's.

b) binary dependencies.

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

Hard for NP: We show two reductions:

a) msd's: reduction from TABLEAUX CONTAINMENT [ASU].

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

Theorem 5. Let d and d` be total mod's. The implication problem of type (d` ; d) is:

a) NP-complete, if d and d` are:

1) many sorted egd's, or

2) binary egd's.

b) NP-hard, if d and d` are:

1) many sorted tgd's, or

2) binary tgd's.

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

a) 1) 3-SATISFIABILITY [Cook],

2) CLIQUE [Ka],

b) 1) EXACT COVER BY 3-SETS [Ka],

2) CLIQUE [Ka]. <>

Consider now a binary many-sorted mod d. Clearly, if d is a pd then it is trivial.

Theorem 6. The implication problem problem of type (binary many sorted mod's ; binary msd's), is equivalent to the finite implication problem and is solvable. <>

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

Lemma 3. In the following cases we have $D \not\models d$ (for a nontrivial d):

a) D is a set of tgd's and d is a egd (also $D \not\models_f d$).

b) D is a set of pd's and d is a td.

c) $D = \{d'\}$, where d' is a partial many sorted mod, and d is a td(also $D \not\models_f d$). <>

Our last solvability result in this section relies on the observation that any decision problem with a finite number of instances is solvable. That is, there exists an algorithm which

gives the right answer to any instance of the problem. this does not mean that the algorithm can be constructively specified.

Theorem 7. Let C be a class of n-ary dependencies, such that for some p_0,q_0 every dependency in C is equivalent to a dependency $\forall^* \exists^* (A_1/\backslash.../\backslash A_p \rightarrow B_1/\backslash.../\backslash B_q)$, where $p \leq p_0$ and $q \leq q_0$, then the implication and the finite implication problems of type (C ; C) are solvable. <>

Corolarry. For a fixed collection of attributes, the implication and the finite implication problems for embedded dependencies of the following types are solvable: functional, multivalued, join, full join [Sc], simple tableaux [BV]. <>

We conclude this section by showing how, in some cases, we can eliminate egd's from consideration. Let d be the egd $\forall y_1 ... \forall y_k (A_1/\backslash.../\backslash A_p \rightarrow y_g = y_h)$. Let A denote the predicated formula $R(y_{k+1},...,y_{k+n})$, 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 $\{\forall y_1 ... \forall y_{k+n} (A_1/\backslash.../\backslash A_p/\backslash A(m/y_g) \rightarrow A(m/y_h)) \mid 1 \leq m \leq n\}$, D_2 is defined similiarly, with g and h interchanged, and d' is taken to be the union of D_1 and D_2. We denote by D' the result of replacing each egd d in the set D by 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 $D' \models_f d$. <>

5. UNSOLVABILITY RESULTS

The main result of this section is:

Theorem 8. The implication and the finite implication problems are unsolvable. <>

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

Let L_{id} be the language of first order logic with identity, with finitely many function symbols and 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_{id}. A conditional equation is a sentence $\forall y_1 \ldots \forall y_k(s_1 = t_1 \wedge \ldots \wedge s_{m-1} = t_{m-1} \rightarrow s_m = t_m)$, where $s_1, t_1, \ldots, s_m, t_m$ are terms of L_{id}. Equational logic is a fragment of first order logic, in which equations and conditional equations are the only admitted sentences. Equational logic was originated by [Birk] and has experienced vigorous grouth in the last few years. [Tar] is a survey of this subject area.

Let L_2 be L_{id} with one binary function symbol g. The depth $|t|$ of a term t of L_2 is the maximum depth of nesting in the term, and is defined by:

$|v| = 0$, and

$|g(t_1,t_2)| = \max(|t_1|,|t_2|) + 1$.

An equation $\forall y_1 \ldots \forall y_k(s = t)$ is simple if $|s| > 1$ and $|t| = 0$. A conditional equation is simple if it is of the form $\forall y_1 \ldots \forall y_k(e(1) \wedge \ldots \wedge e(m-1) \rightarrow e(m))$, where e(i) is $g(v_i^1, v_i^2) = v_i^3$.

Lemma 5. The following holds for L_2:

a) For every simple equation we can effectively construct an equivalent simple conditional equation.

b) for every conditional equation we can effectively construct an equivalent simple conditional equation. <>

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

<u>Theorem 9</u>. [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. <>

<u>Corolarry</u>. The above problems are unsolvable even for simple equations.

<u>Proof</u>. Let $\forall y_1 \ldots \forall y_k (s = t)$ be the given equation. Without loss of generality assume that $|s| \geq |t|$. If the equation is not simple then either $|s| \leq 1$ or $|t| \geq 1$. In each of the possible cases either the equation has a non-trivial finite moedl, or it has no non-trivial model. <>

<u>Corolarry</u>. The above probelms are unsolvable even for simple conditional equations.

<u>Proof</u>. By Lemma 5. <>

Equations can be coded by dependencies by replacing function symbols by their representing predicates. Let $U = \langle A,g\rangle$ be a structure for L_2, i.e., U is a <u>groupoid</u>. The <u>representing relation</u> 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 ternaty 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 (universal quantifiers are omitted):

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

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

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

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

<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 10</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 spring board 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: $G(y_1,y_2,y_3) \wedge G(y_1,y_4,y_5) \rightarrow G(y_1,y_2,y_4))$.

Theorem 11. 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) $\{d \mid Gb \cup \{d\} \models_f T2\}$.

Proof. Observe that a groupoid is trivial 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 10 and Lemma 4.

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 depndencies implied by Gc is not recursive.

A group 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 expresed 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. [Bo1]) in the formulation of [McKi].

Theorem 12. The set of conditional equations which holds in all groups is not recursive. <>

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

Theorem 13. The following set of ternary tuple generating mod's is not recursive: $\{d \mid Gc \models d\}$. <>

Remark. In a similiar manner we could reduce other group-theoretical decision problems, e.g., the triviality or the commutativity problem for finitely presented groups [Ra], to the implication problem for dependencies.

6. MORE UNSOLVABILITY RESULTS

In the last section we showed that the implication and the finite implication problems are unsolvable. In this section we show that we can restrict the type of this problems while retaining their unsolvability.

We define several types:

Type 1: (Dep ; egd's),

Type 2: (tgd's ; tgd's),

Type 3: (mod's ; egd's),

Type 4: (tuple generating mod's ; total tuple generating mod's).

From Theorem 11 we get:

Theorem 14. The (finite) implication problem of types 3 and 4 for L(3) is unsolvable. <>

Any ternary relation G can be represented by a unary relation G_1 and a binary relation G_2 such that $\langle x,y,z\rangle \in G$ iff $\{x,y,z\} \subseteq G_1$ and for some u,v,w we have $\{\langle u,x\rangle,\langle v,y\rangle,\langle w,z\rangle,\langle u,v\rangle,\langle v,w\rangle\} \subseteq G_2$. Thus, we get:

Theorem 15. The (finite) implication problem of types 1 and 2 for L(1,2) is unsolvable. <>

Let $L_{1,1}$ be L_{id} with two unary function symbols f and h.

Lemma 7. [Mal] The following problem is unsolvable:

let $Eq_1,\ldots,Eq_n$ be equations of $L_{1,1}$, decide whether $\{Eq_1,\ldots,Eq_{n-1}\} \models Eq_n$. <>

A unary function can be represented by a binary relation. By

expressing equations as dependencies we get:

Theorem 16. The implication problem of types 3 and 4 for L(2,2) is unsolvable. <>

To extend our unsolvability results to msd's we look for a representation scheme of a ternary relation such that the dependencies of L(3) can be expressed by msd's. Unsolvability then follows from Theorem 14. We can do that either by using more than one relation or by using individual constants. Since we want our dependencies to be many sorted, a constant may appear only in one argument position. We denote by $L(n:j_1,\ldots,j_n)$ a language with an n-ary predicate symbol with j_i constants which may appear in its i-th argument position, $1 \leq i \leq n$. E.g., L(3:3,0,0) denotes a language with a ternary predicate symbol and three constants which may appear in its first argument position.

Let Type i` be Type i, i=1,...,4, as defined above with the additional requirement that all dependencies be many sorted.

Theorem 17. The (finite) implication problem of types 1` and 2` for L(3,3) is unsolvable. <>

Proof. We represent the ternary relation G by two ternary relations G_1 and G_2 in the following way. Each element a is replaced by three elemnents a_1, a_2 and a_3. Now $\langle a,b,c\rangle \in G$ iff $\langle a_1,b_2,c_3\rangle \in G_1$ and $\{\langle a_1,a_2,a_3\rangle,\langle b_1,b_2,b_3\rangle,\langle c_1,c_2,c_3\rangle\} \subseteq G_2$. <>

Corolarry. The (finite) implication problem of types 1` and 3` for L(4:2,0,0,0) is unsolvable. <>

Proof. We combine G_1 and G_2 to one relation using constants to "mark"

the tuples. <>

By different representation schemes we get:

Theorem 18. The (finite) implication problem of types 1` and 2` for the following languages is unsolvable:

a) L(3:3,0,0),

b) L(3:1,1,0),

c) L(2:2,3)

d) L(3,1,1),

e) L(2,1,1,1,1,1),

f) L(2,2,1). <>

By Lemma 2, the implication problem is recursively enumerable. The following theorem asserts that the implication problem has an arbitrary degree of unsolvability.

Theorem 19. For any recursively enumerable degree of unsolvability Δ, there exist a recursive class of ternary mod's C, such that the implication problem of type (Gc ; C) is of degree $\geq \Delta$.

Proof. The claim follows from a similiar result for the word problem for groups [Bo2]. <>

The meta implication problem is to decide for given recursive classes of dependencies C_1 and C_2 whether the set $\{d \mid d \in C_2$ and $C_1 \models d\}$ is recursive.

Theorem 20. The meta implication problem is unsolvable.

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

7. IMPLICATION VS. FINITE IMPLICATION

Combining the unsolvability results of Section 5 with Lemma 2 we get:

Theorem 21. The following sets are not recursively enumerable:

a) $\{\langle D,d\rangle \mid D \models_f d\}$

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

Note that we can sharpen this theorem by using the results of Section 6. 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 and a formal system for implication does exist [BV,SU].

By the corolarry of Lemma 2, $\models$ and $\models_f$ are not equivalent for dependencies. That is, there exist a set of dependencies D and a dependency d such that $D \models_f d$ but $D \not\models d$. We demonstrate it for the dependencies of L(3) and L(2,2).

For the first example we use a result from group theory. Relevant definitions can be found in any standard textbook, e.g., [Rot].

Lemma 8. [Hig] Let G be a group generated by a,b,c,d and the defining relations: $\{ba=ab^2, cb=bc^2, dc=cd^2, ad=da^2\}$, then G is infinite, and it has no finite homomorphic image, except for the trivial one. <>

Let H be the sentence

H: $(\forall a)(\forall b)(\forall c)(\forall d)(ba=ab^2/\backslash cb=bc^2/\backslash dc=cd^2/\backslash ad=da^2 \rightarrow a^2=a)$. By Lemma 8, H holds in all finite groups, but there is an infinite group in which it does not hold. That is (recall the group axioms from Section

5),

{H1,H2,H3} $\models_{\bar{f}}$ H but {H1,H2,H3} $\not\models$ H.

Translation to ternary mod's is left to the reader.

For the second example we use the following lemma.

Lemma 9. [Bu] A structure U = <A,f,h> for $L_{1,1}$ which is a model of the equations:

E1: $f(h(f(h^2(x))))=x$

E2: $f(h(f^2(h^2(x))))=f(h(f^2(h^2(y))))$,

is either infinite or trivial. <>

That is, {E1,E2} $\models_{\bar{f}} \forall x \forall y(x=y)$ but {E1,E2} $\not\models \forall x \forall y(x=y)$. Translation to mod's of L(2,2) is left to the reader.

We show now that $\models$ and $\models_{\bar{f}}$ are not equivalent even for binary depndencies, 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: $R(y_1,y_2)/\backslash R(y_2,y_3)\to R(y_1,y_3)$,

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

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

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

Lemma 10.

a) $\{d_1,d_2\} \models_{\bar{f}} d_4$ but $\{d_1,d_2\} \not\models d_4$,

b) $\{d_1,d_2,d_3\} \models_{\bar{f}} d_5$ but $\{d_1,d_2,d_3\} \not\models d_5$. <>

Similiar to the meta implication problem is the implication equivalence problem: to decide for given recursive classes of

dependencies C_1 and C_2 whether for all d $\in C_2$, $C_1 \models_f d$ iff $C_1 \models d$.

Theorem 22. The implication equivalence problem is unsolvable.

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

8. A REDUCTION CLASS

In this section we show that the class of tuple generating cod's and ood's is a reduction class for implication and finite implication in the following sense.

Theorem 23. Let D be a set of dependencies and d a dependency. We can effectively construct a set D' consisting of tuple generating ood's and one tuple generating cod, and a tuple generating ood d', such that $D \models d$ iff $D' \models d'$ and $D \models_f d$ iff $D' \models_f d'$. <>

The reduction proceeds in three steps. First, identity is replaced by a new binary predicate symbol; secondly, the two predicate symbols are replaced by a single predicate symbol; and finally, we reduce the problem to that specified in the theorem.

7.1 Elimination of Identity

Let D and d be given. If d is a tgd, we can apply Lemma 5 to eliminate identity. If d is an egd we use a variant of the standard technique for elimination of identity [DG], and replace each identity formula $v_i=v_j$ in D and d by $I(v_i,v_j)$, where I is a new binary predicate symbol. We want I to be an equivalence relation and have the

substitutivity property. To this end we add to D the following dependencies:

I1: $\{ R(y_1,\ldots,y_n)\rightarrow I(y_i,y_i) \mid 1 \leq i \leq n \}$

I2: $I(y_1,y_2)\rightarrow I(y_2,y_1)$

I3: $I(y_1,y_2)/\backslash I(y_2,y_3)\rightarrow I(y_1,y_3)$

I4: $R(y_1,\ldots,y_n)/\backslash I(y_1,y_{n+1})/\backslash\ldots/\backslash I(y_n,y_{n+n})\rightarrow R(y_{n+1},\ldots,y_{n+n})$.

7.2 Back to One Predicate Symbol

We can replace R and I by a single n+2-ary predicate symbol P, with the intention that $P(a_1,\ldots,a_n,a_{n+1},a_{n+2})$ iff $R(a_1,\ldots,a_n)$ and $I(a_{n+1},a_{n+2})$. To this end we add the following dependency to D:

$P(y_1,\ldots,y_{n+2})/\backslash P(y_{n+3},\ldots,y_{2n+4})\rightarrow P(y_1,\ldots,y_n,y_{2n+3},y_{2n+4})$.

Appropriate changes must be done to all other dependencies in D and d.

7.3 Reducing to Ood's and One Cod.

Let $r = \max \{ \max(p,q) \mid \forall^*\exists^*(A_1/\backslash\ldots/\backslash A_p\rightarrow B_1/\backslash\ldots/\backslash B_q) \in D \cup \{d\} \}$. The basic idea is to replace r tuple in P by one tuple in $Q=P^r$. We show an example for r=2. First we add the following dependencies to D (let m=n+2):

1) $Q(y_1,\ldots,y_{2m})/\backslash Q(y_{2m+1},\ldots,y_{4m})\rightarrow Q(y_1,\ldots,y_m,y_{3m+1},\ldots,y_{4m})$

2) $Q(y_1,\ldots,y_{2m})\rightarrow Q(y_{m+1},\ldots,y_{2m},y_1,\ldots,y_m)$.

This dependencies ensure that $Q=P^2$ for some P. Now we replace a dependency

$\forall^*\exists^*P(v_1,\ldots,v_m)/\backslash P(v_{m+1},\ldots,v_{2m})\rightarrow P(v_{2m+1},\ldots,v_{3m})$ by

$\forall^*\exists^*Q(v_1,.,v_m,v_{m+1},\ldots,v_{2m})\rightarrow Q(v_1,\ldots,v_m,v_{2m+1},\ldots,v_{3m})$.

7.4 Solvability and Unsolvability

The results of Section 5 combined with Theorem 23 yields:

Theorem 24. The implication and the finite implication problems of type (tuple generating ood's + one tuple generating cod ; tuple generating ood's) are unsolvable.

The unsolvability stems from mixing ood's and cod's since each class by itself is solvable. In fact, even broader classes are solvable.

Theorem 25. The implication and the finite implication problems of the following types are equivalent and solvable:

a) (cd's ; mod's),

b) (omd's ; mod's). <>

9. CONCLUDING REMARKS

In the preceeding sections we have demonstrated the unsolvability of the implication and the finite implication problems for dependencies, pointed out some solvable classes, and provided some complexity bounds. Several questions, however, remain open.

By theorem 14, the implication problem for mod's of L(2,2) is unsolvable. The solvability of the finite implication problem for this class is open. Observe that unlike part (a) of Theorem 7, part (b) does not hold for $L_{1,1}$, and the set of equations in $L_{1,1}$ which have a

non-trivial finite model is recursive [Bu].

The solvability of our decision problems for the class of binary dependencies is also open. Since the theory of one function is decidable [Ehr], we can not hope to settle this question by the technique of this paper. Observe that by Lemma 10, $\models$ and $\models_{\bar{f}}$ are not equivalent for binary dependencies.

Our unsolvability proof in Section 5 uses two existential quantifiers (one in G1 and the other in G2). Can we extend this result to the case that D contains a single existential quantifier? or is this restricted class solvable? Observe that $\models$ and $\models_{\bar{f}}$ are obviously not equivalent even in our restricted case, as is evident from Lemma 10. For msd's, however, this restriction yields solvability.

Theorem 26. Let D be a set of msd's such that for some j, $1 \leq j \leq n$, all existentially quantified variables occur as the j-th argument of predicated formulas, and let d be any msd, then $D \models d$ iff $D \models_{\bar{f}} d$. <>

Some restricted classes of msd's of L(n) are known to be solvable [BV,SW], however, the solvability of the whole class is still open, and is not amenable to the technique of this paper.

Finally, note that for all known solvable classes of dependencies $\models$ is equivalent to $\models_{\bar{f}}$. It would be interesting to find a solvable class for which it is not the case

RELATED WORK

The unsolvability of the implication and the finite implication problems for dependencies has been proven independently by [CLM], by using 6-ary dependencies to encode the halting problem for two-counter machines. They have also shown that the implication problem for td's is logspace complete in EXPSPACE.

ACKNOWLEDGEMENTS

We would like to thank J.A. Makowsky for fruitful discussions which set the initiative to this paper.

REFERENCES

[ABU] Aho, A.V., Beeri, C., Ullman, J.D.: The theory of joins in relational databases. ACM Trans. on Database Systems 4:3(Sept. 1979), pp. 297-314

[BB] Beeri, C., Bernstein P.A.: Computational problems related to the design of normal form relational schemas. ACM Trans. on Database Systems 4:1 (March 1979), pp. 30-59.

[Beer] Beeri, C.: On the membership problem for multivalued dependencies. to appear in ACM Trans. on Database Systems.

[Birk] Birkhoff, G.: On the structure of abstract algebras, Proc. Cambridge Phil. Soc. 31(1935), pp. 433-454.

[Bo1] Boone, W.W.: The word problem. Ann. of Math. 70(1959), pp. 207-265.

[Bo2] Boone, W.W.: Word problems and recursively enumerable degrees of unsolvability - a sequel on finitely presented groups. Ann. of Math. 84(1960), pp.49-84.

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

[Bu] Burris, S.: Models in equational theories of unary algebras. Algebra Universalis 1(1972), pp. 386-392.

[BV] Beeri, C., Vardi, M.Y.: unpublished results

[Ch] Church, A.: A note on Eintscheidungsproblem. J. of Symbolic Logic 1(1936), pp.40-41.

[CLM] Chandra, A.K., Lewis, H.R., Makowsky, J.A.: Embedded implicational dependencies and their inference problem. to appear.

[Cook] Cook, S.A.: The complexity of theorem proving procedures. Proc. 3rd Ann. ACM Symp. on Theory of Comput., 1971, pp. 151-158.

[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.

[DG] Dreben, B., Goldfarb, W.D.: The decision problem - solvable classes of quantificational formulas. Addison-Wesley, Reading, 1979.

[Ehr] Ehrenfeucht, A.: Decidability of the theory of one function. Notices of the American Math. Soc. 6(1959), p. 268

[Fag] Fagin, R.: Multivalued dependencies and a new normal form for relational databases. ACM Trans. on Database Systems 2:3(Sept. 1977), pp. 262-278.

[Hig] Higman, G.: A finitely generated infinite simple group. J.

London Math. Soc. 26(1951), pp. 61-64.

[JP] Janssens, D., Paredaens, J.: General dependencies. Workshop on Formal Bases for Data Bases, 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.

[Le] Lewis, H.R.: The complexity of solvable cases of the decision problem for the predicate calculus. Proc. 19th Ann. Symp. on Found. of Computer Science, 1978, pp. 35-47.

[Mal] Malc'ev, A.I.: Identical relations on varieties of quasigroups. Mat. Sb. 69(111)(1966), pp.3-12, translated in American Math. Soc. translation, Series 2, Vol 82, 1969, pp. 225-236.

[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.

[McKi] McKinsey, J.C.C.: The decision problem for some classes of sentences without quantifiers. J. of Symbolic Logic 8(1943), pp.61-76.

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

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

[Nico1] Nicolas, J.M.: Mutual dependencies and some results on undecomposable relations. Proc. 4th Int'l Conf. on VLDB, 1978, pp.360-367.

[Nico2] Nicolas, J.M.:First order logic formalization for functional,

multivalued and mutual dependencies. Proc. ACM-SIGMOD Int'l Conf. on Management of Data, 1978, pp.40-46.

[Pa] Paredaens, J.: Transitive dependencies in a database scheme. Report R387, MBLE Research Lab, Brussels, Feb. 1979.

[Pl] Plaisted, D.A.: Complete problems in the first order calculus. Research Report UIUCDCS-R-79-978, Dept. of Computer Science, Univ. of Illinois at Urbana-Champaign, July 1979.

[Ra] Rabin, M.O.: Recursive unsolvability of group-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 Foundations of Computer Science, Poland, 1978, Lecture Notes in Computer Science 64, Springer-Verlag, pp. 537-551.

[Rot] Rotman, J.J.: The theory of groups. Allyn and Bacon, Boston, 1965.

[Sc] Sciore, E.: A complete axiomatization of full join dependencies. TR #279, Department of EECS, Princeton University, July 1979.

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

[SW] Sagiv, Y., Walecka, S.: Subset dependencies as an alternative to embedded multivalued dependencies. Technical Report UIUCDCS-R-79-980, University of Illinois at Urbana-Champaign, 1979.

[Tar] Tarski, A.: Equational logic and equational theories of algebras. in Contribution to Mathematical Logic (K. Schutte, ed.), North-Holland, Amsterdam, 1968, pp. 275-288.

[TMR] Tarski, A., Mostowski, A., Robinson, R.M.: Undecidable theories.

North-Holland, Amsterdam, 1953.

[Tr] Trachtenbrot, B.A.: Impossibility of an algorithm for the decision problem in finite classes. Doklad. Akad. Nauk SSSR 70(1950), pp. 569-572, translated in American Math. Soc. Tranlation, Series 2, Vol 23, 1963, pp.1-5.

[Va] Vardi, M.Y.:Inferring multivalued dependencies from functional and join dependencies. Research Report, Dept. of Applied Math., Weizmann Inst. of Science, March 1980.

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