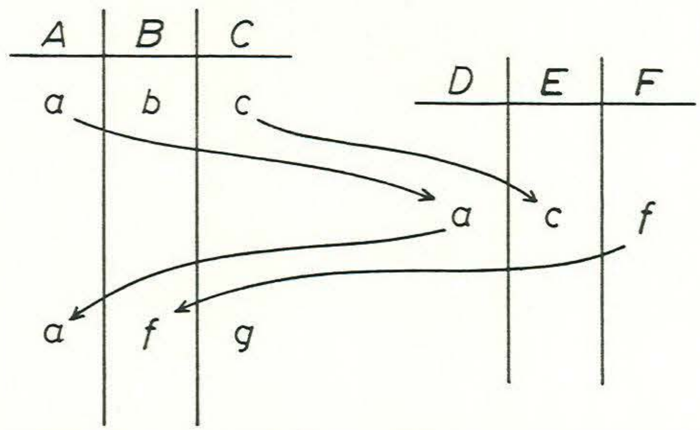
MIT/LCS/TM-235

THE IMPLICATION PROBLEM FOR FUNCTIONAL AND INCLUSION DEPENDENCIES



John C. Mitchell

February 1983

The Implication Problem for Functional and Inclusion Dependencies

John C. Mitchell
Laboratory for Computer Science
Massachusetts Institute of Technology
Cambridge, MA 02139

February 2, 1983

Abstract

There are two implication problems for functional dependencies and inclusion dependencies: general implication and finite implication. Given a set of dependencies $\Sigma \cup \{\sigma\}$, the problems are to determine whether σ holds in all databases satisfying Σ or all finite databases satisfying Σ. Contrary to the possibility suggested in [5], there is a natural, complete axiom system for general implication. However, a simple observation shows that both implication problems are recursively unsolvable. It follows that there is no recursively enumerable set of axioms for finite implication.

Categories and Subject Descriptors: H.2.1 **[Database Management]**: Logical Design

General Terms: Theory

Additional Keywords and Phrases: relational database, inclusion dependency, functional dependency, complete axiomatization, undecidability, finite implication

The author is supported by a fellowship from the National Science Foundation.

1. Introduction

Functional dependencies and inclusion dependencies are simple and important integrity constraints for relational databases. Both kinds of dependencies are first-order sentences that are useful for describing or specifying database designs.

Functional dependencies have been studied extensively (e.g. [1, 7, 2]). The ubiquitous example of a functional dependency is the typical correspondence between employees and managers. Since every employee has precisely one manager, any database of office personnel contains a function between its employees and managers. In other words, the *attribute* EMPLOYEE functionally determines the attribute MANAGER. Formally, this is written EMPLOYEE $\rightarrow$ MANAGER. For functional dependencies, finite and general implication coincide. Implication for functional dependencies has a well-known axiomatization [1] and an efficient decision procedure [2]. Inference rules and decision procedures have also been developed for functional dependencies in combination with various other dependencies (e.g. [3, 11, 22]).

Although inclusion dependencies are common in database practice [4, 6, 8, 10], the theoretical properties of inclusion dependencies have received relatively little attention until quite recently. An inclusion dependency arises in the EMPLOYEE and MANAGER database. In a typical corporation, every MANAGER is also and EMPLOYEE. Hence the set of employees in any office database will include the set of managers in the database. This inclusion dependency is written MANAGER $\subseteq$ EMPLOYEE. As for functional dependencies, implication and finite implication coincide for inclusion dependencies. Recent theoretical papers on inclusion and functional dependencies include [5] and [16]. In particular, [5] describes the interaction between functional and inclusion dependencies and discusses previous work by other authors. In [5], a straightforward set of inference rules for inclusion dependencies is presented and proved complete. Furthermore, the implication problem for inclusion dependencies is shown to be PSPACE-complete (cf. [12]).

General implication and finite implication differ when functional dependencies and inclusion dependencies are considered together [5]. Since we will most often be concerned with general implication, the term implication will refer to general implication unless otherwise specified.

Implication for functional and inclusion dependencies has an unusual property, as shown by [5]. A dependency σ follows from a set of dependencies Σ by *k-ary implication* if there is some subset of k dependencies from Σ that implies σ. In [5], the authors show that for every (sufficiently large) integer k, there is a set of functional and inclusion dependencies which is closed under k-ary implication but not closed under implication. This theorem suggests that there is no natural, complete axiom system for functional dependencies and inclusion dependencies together. This is because a single inference rule generally yields a single consequence of k antecedents. Furthermore, there is some fixed upper bound on k for the entire system. Thus most axiom systems are complete only for k-ary implication. Since k-ary implication for functional and inclusion dependencies differs from implication, no straightforward, simply presented axiom system of the usual sort is likely to be complete.

This paper presents axioms and inference rules that are complete.[1] The rules differ from those considered by [5] in two respects. A minor difference is that inclusion dependencies are allowed to contain sequences of attributes with duplicate elements. This seems natural, and gives inclusion dependencies slightly greater expressive power. Specifically, equality may be expressed using inclusion dependencies. More importantly, one inference rule yields dependencies which mention attributes that are not used in the hypotheses. This *attribute introduction rule* distinguishes the inference system from the variety considered by [5]. The inference rules of the system are all "k-ary" in that each rule yields a single new consequent by inspection of at most three antecedents. However, *attribute introduction* is not sound in the usual sense. The inference system is also "universe unbounded" (cf. [21]) since the set of attributes used in a single deduction may be arbitrarily large.

The attribute introduction rule allows new attribute names representing "derived" attributes to be introduced into deductions. An example will illustrate the intuitive interpretation for the new attribute names. Consider a database of employees, managers and salaries. We can abbreviate the names of the employee, manager and salary attributes to EMP, MGR and SAL. Each *tuple*, or row in the database "table" lists an employee, his or her manager, and the employee's salary. Since every employee has a single salary, we have EMP $\rightarrow$ SAL. In addition, since every manager is an employee,

---

[1] Of course, since finite implication differs from general implication, the rules are not complete for finite implication.

MGR $\subseteq$ EMP. As a consequence, the database associates a single salary with each *manager*. To find the salary of a manager, say Bob, we find a tuple listing Bob as an employee, then look up the salary given in that tuple. Since MGR $\subseteq$ EMP, we know that Bob is somewhere in the relation as an employee. Because EMP $\rightarrow$ SAL, the salary we find is uniquely determined. To describe the fact that MGR uniquely determines "manager salary", we could add a new attribute to the database MSAL for managers salaries and write MGR $\rightarrow$ MSAL. The entries in the new column MSAL, with

$$\text{MGR, MSAL} \subseteq \text{EMP, SAL}$$

are completely determined by the employee, manager and salary entries in the original database. As shown in Section 3, this follows from the fact that

$$\text{MGR} \subseteq \text{EMP} \quad \text{and} \quad \text{EMP} \rightarrow \text{SAL}.$$

The attribute introduction rule simplifies reasoning about functional and inclusion dependencies by introducing new attributes like MSAL which can be thought of as attributes whose entries are computed or derived from the original portion of the database. Intuitively, the main use of new attributes lies in the possibility of proving that they are equivalent to original attributes.

One way of viewing the new attribute MSAL is as an abbreviation for an *attribute expression* in an extended dependency language. This perspective will lead us to a simple proof of undecidability for both the finite and general implication problems. Any relation satisfying EMP $\rightarrow$ SAL contains a function between its employee entries and its salary entries. We could name this function by putting braces {,} around the functional dependency and write

$$\text{SAL} = \{\text{EMP}\rightarrow\text{SAL}\}(\text{EMP})$$

to mean that the salary entry in any tuple (or "row") of the database is the result of applying the {EMP$\rightarrow$SAL} function to the employee entry in that tuple. This function {EMP$\rightarrow$SAL} is related to the new attribute MSAL since it is the "rule" for computing MSAL entries, i.e.

$$\text{MSAL} = \{\text{EMP}\rightarrow\text{SAL}\}\,(\text{MGR}).$$

We know that each manager is in the domain of {EMP$\rightarrow$SAL} since MGR $\subseteq$ EMP. Instead of using new attribute names like MSAL in deductions, we could use expressions like {EMP$\rightarrow$SAL}(MGR).

The use of attribute expressions leads one to thinking of inclusion dependencies as statements about functions named by functional dependencies. For example, if we assume that A$\rightarrow$B and C$\rightarrow$D, then the dependencies EF$\subseteq$AB and EF$\subseteq$CD can be interpreted as statements about the functions {A$\rightarrow$B} and $\{C \rightarrow D\}$. These two inclusion dependencies imply that

$$F = \{A \rightarrow B\}(E) \quad \text{and} \quad F = \{C \rightarrow D\}(E).$$

This forces $\{A \rightarrow B\}$ and $\{C \rightarrow D\}$ to agree on all entries in the E column of the database. If the domain of $\{C \rightarrow D\}$ is in the range of $\{A \rightarrow B\}$, there are also dependencies which express properties of the composition $\{A \rightarrow B\} \circ \{C \rightarrow D\}$.

In a sense, it is the fact that dependencies make statements about functions which makes functional dependencies combined with inclusion dependencies intractable. The implication problem for monoids (word problem) can be reduced to the general implication problem for functional and inclusion dependencies by translating equations between compostions of functions into dependencies. The same translation also reduces implication over finite monoids to the finite implication problem for dependencies. Since the implications valid over all finite monoids are not recursively enumerable [13, 14], there is no complete, recursively enumerable axiomatization for finite implication of inclusion dependencies and functional dependencies.[2]

2. Databases and Dependencies

Formally, a *relational data base scheme* is a set $\mathfrak{R}$ of *relation names.* Each relation name $R \in \mathfrak{R}$ has associated attributes R[1], R[2], ... In practice, attributes have meaningful names like EMPLOYEE, MANAGER, etc. but for the purposes of this paper the integers 1, 2, ... do just fine. Infinitely many attributes are used so that attribute introduction is easy to formalize. A *relation* r is a set of tuples and a *database* for a scheme $\mathfrak{R}$ is a nonempty relation r for each $R \in \mathfrak{R}$. A tuple $t \in r$ is a sequence of *entries* $\langle a_1, a_2, \ldots \rangle$. We write t[i] to denote the i-th entry of t. If X is a finite sequence of attributes $\langle X_1, X_2, \ldots \rangle$, then t[X] denotes the sequence of entries $\langle t[X_1], t[X_2], \ldots \rangle$ and |X| denotes the length of X. Note that some attribute may appear more than once in X. We write r[X] for $\{t[X] \mid t \in r\}$. A relation is *finite* if it consists of finitely many tuples and a database is finite if it consists only of finite relations.

Following common convention, capital letters from the beginning of the alphabet A, B, C, ... will be used to denote single attributes while capital letters from the end of the alphabet U, V, W, X, ... will

[2] The author has been informed that these undecidability results have also been obtained independently by Chandra and Vardi, although presumably by different methods [9].

denote nonempty sequences of attributes. Lowercase s and t, possibly with subscripts, will denote tuples and r a relation (set of tuples).

A relation r' is an A-*variant* of r if there is a bijection f from r to r' such that for all $t \in r$ and all attributes $B \neq A$, $f(t)[B] = t[B]$."

A *functional dependency* is an assertion of the form $R{:}X \rightarrow Y$, where X and Y are nonempty sequences of attributes. A relation r satisfies $R{:}X \rightarrow Y$ if, for any tuples s and t in r, $s[X] = t[X]$ implies $s[Y] = t[Y]$. An *inclusion dependency* is an assertion of the form $R[X] \subseteq R'[Y]$. A database $\langle r, r', ... \rangle$ satisfies $R[X] \subseteq R'[Y]$ if $r[X] \subseteq r'[Y]$. It is occasionally convenient to write

$$\Sigma \models \sigma$$

if every database satisfying Σ also satisfies σ. The notation $\Sigma \models_{\text{finite}} \sigma$ means that σ holds in every finite database which satisfies Σ.

To keep the notation simple, all inference rules presented in this paper are written for functional and inclusion dependencies which mention only one relation. Consequently, relation names are omitted from dependencies. All the rules can be rewritten to apply to arbitrary database schemes. The completeness proof in Section 4 is also easily extended to arbitrary schemes.

3. Attribute Introduction Rules

The *attribute introduction inference system* combines several known rules for functional dependencies or inclusion dependencies together with an equality rule and three new rules involving both kinds of dependencies.[3] The salient new rule of the system is the attribute introduction rule,

From $U \subseteq V$ and $V \rightarrow B$ derive $UA \subseteq VB$.

This rule is not sound in the usual sense since there exist relations satisfying $U \subseteq V$ and $V \rightarrow B$ which do not satisfy $UA \subseteq VB$. However, with the proper definition of proof, all proofs of the system will be sound. Proofs are defined following the presentation of the axioms and rules.

---

[3] Two combined rules, listed as FI1 and FI2 below, were discovered independently by the author and by Casanova, Fagin and Papadimitriou. The soundness proofs for these rules are Propositions 4.1 and 4.2 in the IBM Technical Report version of [5]. The functional dependency rules F1-F3 are from [1] and the inclusion dependency rules I1-I3 from [5]as published in the 1982 ACM PODS Conference.

Functional Dependencies

(Reflexivity Axiom)
F1. $X \rightarrow Y$ if all attributes in Y appear in X,

(Augmentation)
F2. From $X \rightarrow Y$ derive $XW \rightarrow YZ$ when all attributes in Z appear in W

(Transitivity)
F3. From $X \rightarrow Y$ and $Y \rightarrow Z$ derive $X \rightarrow Z$,

(Permutation and Redundancy)
F4. From $X \rightarrow Y$ derive $U \rightarrow V$, where U and V list precisely the same attributes as X and Y, respectively,

Inclusion Dependencies

(Reflexivity Axiom)
I1. $X \subseteq X$

(Permutation, Projection and Redundancy)
I2. From $A_1,...,A_n \subseteq B_1,...,B_n$ derive $A_{i_1},...,A_{i_k} \subseteq B_{i_1},...,B_{i_k}$, where $1 \leq i_j \leq n$ for all j,

(Transitivity)
I3. From $X \subseteq Y$ and $Y \subseteq Z$ derive $X \subseteq Z$

(Substitutivity of Equivalents)
I4. From $AB \subseteq CC$ and σ derive τ, where τ is obtained from σ by substituting A for one or more occurrences of B

Functional and Inclusion Dependencies

(Pullback)
FI1. From $UV \subseteq XY$ and $X \rightarrow Y$ derive $U \rightarrow V$, where $|X| = |U|$,

(Collection)
FI2. From $UV \subseteq XY$, $UW \subseteq XZ$ and $X \rightarrow Y$ derive $UVW \subseteq XYZ$, where $|X| = |U|$,

(Attribute Introduction)
FI3. From $U \subseteq V$ and $V \rightarrow B$ derive $UA \subseteq VB$.

In an application of FI3 where $U \subseteq V$ and $V \rightarrow B$ are used to derive $UA \subseteq VB$, the attribute A is called the *new* attribute of the proof step. In order for the rules above to be sound, we need to restrict the choices of new attributes in proofs. Formally, proofs are defined as follows. Let Σ denote a set of functional dependencies and inclusion dependencies. A *proof from* Σ is a sequence of dependencies $\langle\sigma_1,...,\sigma_n\rangle$ such that

(i) each σ_i is either an element of Σ, an instance of F1 or I1, or follows from one or more of the preceding dependencies $\sigma_1,...,\sigma_{i-1}$ by a single rule,

(ii) if σ_i follows from preceding dependencies by attribute introduction (rule FI3) then the new attribute of σ_i must not appear in Σ or $\sigma_1,...,\sigma_{i-1}$

An inclusion or functional dependency σ is *provable from* Σ, written $\Sigma \vdash \sigma$ if there is some proof $\langle\sigma_1,...,\sigma_n\rangle$ from Σ with $\sigma = \sigma_n$ and such that no attributes in σ are new in $\sigma_1,...,\sigma_n$. The completeness theorem can now be stated.

Theorem 1: Let $\Sigma \cup \{\sigma\}$ be a set of functional dependencies and inclusion dependencies. Then $\Sigma \models \sigma$ iff $\Sigma \vdash \sigma$.

An induction on the lengths of proofs $\langle\sigma_1,...,\sigma_n\rangle$ from Σ shows that if a relation r satisfies Σ, then there is a relation r' which differs from r only on new attributes of the proof and which satisfies each σ_i. It follows that the inference system is sound. The only complicated cases of the induction are when FI3 is used, and possibly the equality rule I4. The attribute introduction case is discussed below and equality subsequently. The full proof of soundness is left to the reader.

The new attribute A in the attribute introduction rule should be thought of as implicitly *existentially* quantified. Attribute introduction yields sound proofs since for every relation r satisfying $U \subseteq V$ and $V \rightarrow B$, there is an A-variant r' of r satisfying $UA \subseteq VB$. The entries in r'[A] are uniquely determined by r[UVB]. Specifically, we can construct r' from r as follows. For any sequence of entries $\langle v_1,...,v_k\rangle \in r[V]$, define $g(v_1,...,v_k)$ by

$\langle v_1,...,v_k, g(v_1,...,v_k)\rangle \in r[VB]$.

Since r satisfies $V \rightarrow B$, this condition defines the function g uniquely. Furthermore, since r satisfies $U \subseteq V$, the projection r[U] is a subset of the domain of g. Using g, we can define r' by

$r' = \{t' \mid t'[A] = g(t'[U])\}$ and $\exists t \in r$ such that $\forall C \neq A$, $t[C] = t'[C]\}$.

Then r' is an A-variant of r and r' satisfies $UA \subseteq VB$. Thus for every r satisfying $U \subseteq V$ and $V \rightarrow B$, there is an A-variant r' satisfying $UA \subseteq VB$.

In [19], a slightly different formulation of the attribute introduction rule is compared to an existential instantiation rule in a natural deduction system for predicate calculus. A sample proof using FI3 and other rules is given at the end of this section.

Repeated Attributes and Equality

A dependency $X \subseteq Y$ or $X \rightarrow Y$ has *repeated attributes* if there is some attribute A that appears at least twice in X or twice in Y. As mentioned in the introduction, equality may be expressed using inclusion dependencies with repeated attributes. Specifically, if any relation satisfies $AB \subseteq CC$, then the A and B entries in any tuple of the relation must be identical. To see why this is so, let t be any tuple in a relation satisfying $AB \subseteq CC$. Then there is some other tuple s in the relation with $t[AB] = s[CC]$, i.e. $t[A] = s[C] = t[B]$. The inclusion $CC \subseteq AB$ does not imply any equality of attributes but does express a nontrivial property of a database. Repeated attributes make no difference for functional dependencies: any functional dependency with repeated attributes is equivalent to one without.

In [5], the authors consider *repeating dependencies* of the form $X=Y$. The repeating dependency $X=Y$ is equivalent to the inclusion dependency $XY \subseteq XX$. However, the inclusion dependency $XX \subseteq XY$ is not equivalent to any set Σ consisting only of inclusion dependencies without repeated attributes and repeating dependencies. A simple modification to the proof presented in [5] extends their "no k-ary axiomatization" theorem to the slightly more powerful dependencies with repeated attributes.

> **Theorem** [Casanova, Fagin, Papadimitriou] **:** For every k, there is a set Σ of inclusion and functional dependencies such that all consequences of every subset of Σ of size k are included in Σ, yet Σ is not closed under implication.

The inclusion dependency rules I1 through I3 are taken from [5] and are shown there to be complete for inclusion dependencies without repeated attributes. Specifically, if Σ is a set of inclusion dependencies without repeated attributes and σ is another such dependency, then Σ implies σ iff σ is provable from Σ by I1, I2 and I3.

It may be shown that I1 through I4 are complete for inclusion dependencies with repeated attributes. A corollary is that no set of inclusion dependencies without repeated attributes implies an inclusion dependency with "nontrivially" repeated attributes. More precisely, if Σ is a set of inclusion dependencies without repeated attributes and Σ implies the inclusion dependency σ, then σ is equivalent to an inclusion dependency without repeated attributes. In contrast, inclusion and functional dependencies together do not share this property. The following example shows that there are sets of functional and inclusion dependencies without repeated attributes that imply dependencies of the form $AB \subseteq CC$.

Example Deduction

Although the results of [5] show that the rules of Theorem 1 cannot be complete without the attribute introduction rule FI3, it is interesting to consider an example which illustrates where FI3 is needed. Let Σ be the following set of hypotheses:

(h1) $C \rightarrow D$

(h2) $AB \subseteq CD$

(h3) $BA \subseteq CD$

(h4) $B \subseteq A$

and let σ be $AB \subseteq BA$. The reader may verify that σ cannot be derived using inference rules other than FI3 by checking all possible deductions (there really are not very many).

Although a little tricky, it is not too difficult to see why Σ implies σ. Consider any tuple t_1 in any relation r satisfying Σ. Suppose $t_1[AB] = \langle a, b\rangle$. Since $B \subseteq A$, there must be a tuple $t_2 \in r$ with $t_2[AB] = \langle b, x\rangle$ for some x. We will see that $AB \subseteq BA$ by determining that x must equal a. Since $AB \subseteq CD$, there must be some tuple t_3 with $t_3[CD] = t_2[AB] = \langle b, x\rangle$. Similarly, from $BA \subseteq CD$ we know that there must be some tuple t_4 with $t_4[CD] = t_1[BA] = \langle b, a\rangle$. But since $C \rightarrow D$ and $t_3[C] = t_4[C]$, it must be that t2-(3)[D] $= t_4[D]$. Thus $x = a$, which proves that σ follows from Σ.

We can prove σ from Σ using the inference rules as follows.

(1) $A \rightarrow B$ from (h1) and (h2) by FI1, Pullback,

(2) $BE \subseteq AB$ by FI3 from (h4) and (1); note that E does no appear in Σ or previously in the proof,

(3) $BE \subseteq CD$ from (2) and (h2) by Transitivity, F3,

(4) $BAE \subseteq CDD$ from (h3), (3) and (h1) by FI2,

(5) $AE \subseteq DD$ by I2,

(6) $BA \subseteq AB$ from (2) and (5) by I4, Substitutivity of Equivalents.

This derivation shows how a new attribute may be introduced and then proved equal to an attribute which appears in the original hypotheses.

4. Completeness

This section proves that the attribute introduction rules are complete; soundness is left to the reader. Let Σ_0 be a set of dependencies and σ a dependency that is not provable from Σ_0 by the attribute introduction rules. Theorem 1 is proved by constructing a relation that satisfies Σ_0 but not σ. The relation is constructed from a larger set of dependencies $\Sigma \supseteq \Sigma_0$ in stages, with a new tuple added at each stage.

A slight inconvenience is that there are two cases: σ may be an inclusion dependency or σ may be a functional dependency. To avoid considering each case separately, we choose three sequences of attributes X_0, Y_0 and Z_0 and construct a relation in which both

$$X_0 \rightarrow Y_0 \quad \text{and} \quad X_0 \subseteq Z_0$$

fail. If σ is a functional dependency $X_0 \rightarrow Y_0$, then choose Z_0 to be a sequence of attributes that do not appear in $\Sigma_0 \cup \{\sigma\}$ and with $|Z_0| = |X_0|$. If σ is an inclusion dependency $X_0 \subseteq Z_0$, then let Y_0 be a single attribute which does not appear in $\Sigma_0 \cup \{\sigma\}$. Note that if there is some relation that satisfies Σ_0 but not σ, then there is also a relation that satisfies Σ_0 but neither $X_0 \rightarrow Y_0$ nor $X_0 \subseteq Z_0$. Thus, since the rules are sound, neither $X_0 \rightarrow Y_0$ nor $X_0 \subseteq Z_0$ is provable from Σ_0.

A set of dependencies Σ is *deductively closed* if Σ is closed under all inference rules except FI3 and for all $(U \subseteq V)$, $(V \rightarrow B) \in \Sigma$, there is some attribute A with $(UA \subseteq VB) \in \Sigma$. We need a deductively closed set containing Σ_0 to carry out the construction. Let

$$\Sigma_1 = \Sigma_0 \cup \{X_0 \rightarrow X_0, Y_0 \rightarrow Y_0, Z_0 \rightarrow Z_0\}$$

so that Σ_1 has the same consequences as Σ_0 but also includes all attributes in X_0, Y_0 and Z_0. This is so that any "new" attributes introduced in any proof from Σ_1 will not be attributes which appear in X_0, Y_0 or Z_0. Let $\Sigma \supseteq \Sigma_1$ be deductively closed. (We can construct such a set in stages by adding formulas used in proofs, including those with new attributes, to Σ_0.)

Since all dependencies in Σ_1 are provable from Σ_0, neither $X_0 \rightarrow Y_0$ nor $X_0 \subseteq Z_0$ is an element of Σ. Theorem 1 is proved by constructing a relation that satisfies Σ but does not satisfy σ.

The outline of the construction is as follows. We fix some arbitrary infinite set S and choose elements of S as entries in tuples. In the first stage of the construction, two tuples t_0 and t_1 are chosen so that $X_0 \rightarrow Y_0$ fails in the relation $r_1 = \{t_0, t_1\}$. Then, at stage $k+1$, an additional tuple t_{k+1} is added to the relation produced so far to "help" satisfy some inclusion dependency $U_k \subseteq V_k$ in Σ. This is done in such a way that all functional dependencies in Σ hold at each stage. Furthermore, no inclusion dependency not in Σ will be satisfied. If the relation r_k produced at stage k does not satisfy $U_k \subseteq V_k$, then we pick a tuple t_i with $t_i[U_k]$ not in $r_k[V_k]$. The new tuple t_{k+1} for stage $k+1$ has $t_{k+1}[V_k] = t_i[U_k]$. The other entries in t_{k+1} are chosen according to a "pullback function" described later. The relation r_{k+1} formed at stage $k+1$ is the $r_k \cup \{t_{k+1}\}$. We call $k+1$ the *index* of tuple t_{k+1}, i.e. the number of the stage at which it was added, and call tuple t_i the *predecessor* of tuple t_{k+1}. All entries in t_{k+1} either occur in its predecessor t_i or do not appear in r_k at all. We write $\leq$ for the reflexive and transitive closure of the predecessor relation, i.e. $s \leq t$ if $s = t$ or if there is some sequence of predecessors leading from t back to s.

The final relation $r = \cup_k r_k$ will be shown to satisfy precisely the inclusion dependencies in Σ. This is accomplished using a property (*), described below, which is shown inductively to hold at each stage. Since r will not satisfy $X_0 \rightarrow Y_0$ by choice of t_0 and t_1, and r will not satisfy $X_0 \subseteq Y_0$ since this inclusion dependency is not in Σ, the relation r will not satisfy σ.

Attribute Equivalence and Pullback Function

In the remainder of the proof, with Σ fixed, two sequences of attributes X and Y are said to be *equivalent*, written $X \equiv Y$, if $XY \subseteq XX \in \Sigma$. The equation $X = Y$ is used only to denote that X and Y are *syntactically identical* sequences of attributes. We use $(V)_i$ to denote the i-th attribute appearing in the sequence of attributes V. Thus $(U)_i \equiv (V)_j$ means that Σ contains the inclusion dependency

$AB \subseteq AA$, where A is the i-th attribute in U and B the j-th attribute in V.

A helpful tool in the construction is a *pullback function* p which is used to choose attributes in a consistent manner. A function, rather than a relation, is used to emphasize that identical choices are made in identical situations. For every pair of dependencies $(U \subseteq V)$, $(V \rightarrow B) \in \Sigma$, there is an attribute A with $(UA \subseteq VB) \in \Sigma$. The attribute A is the image of U under the "pullback" of function $V \rightarrow B$ to U. The following lemma show that the "pullback" is unique, modulo equivalence of attributes.

Lemma 1: Let X and Y be any sequences of attributes. Suppose that $(X \subseteq Y)$ is a permutation and projection of both $(U_1 \subseteq V_1)$ and $(U_2 \subseteq V_2)$. If Σ contains the dependencies

$$X \subseteq Y \quad Y \rightarrow B \quad U_1 \subseteq V_1 \quad U_2 \subseteq V_2$$

and B appears in both V_1 and V_2, i.e. $B = (V_1)_j = (V_2)_k$ for some j and k, then $(U_1)_j \equiv (U_2)_k$.

Proof: Let A_1 denote $(U_1)_j$ and A_2 denote $(U_2)_k$. By projection and permutation, we have

$$XA_1 \subseteq YB \quad \text{and} \quad XA_2 \subseteq YB$$

in Σ since Σ is deductively closed. By Collection, $XA_1A_2 \subseteq YBB \in \Sigma$ and so by Projection and Permutation $A_1A_2 \subseteq BB \in \Sigma$. Since $A_1A_2 \subseteq A_1A_2 \in \Sigma$, we conclude $A_1A_2 \subseteq A_1A_1 \in \Sigma$. Thus $A_1 = (U_1)_j \equiv (U_2)_k = A_2$. ∎

Assume that $(U \subseteq V)$, $(V \rightarrow B) \in \Sigma$. Define p(U,V,B) as follows:

(i) If B appears first as the k-th attribute of V, i.e. if $B = (V)_k$ and $B \neq (V)_j$ for all $j < k$, then define $p(U,V,B) = (U)_k$. Note that if $B = (V)_j = (V)_k$, then $(U)_j \equiv (U)_k$.

(ii) If B does not appear in V, then pick any inclusion dependency $(UA \subseteq VB) \in \Sigma$. Since Σ is deductively closed, there is some $(UA \subseteq VB) \in \Sigma$. Define $p(U,V,B) = A$. By Lemma 1, this choice is unique up to attribute equivalence.

We may extend p to a "pullback" function for sequences by

$$(p(U,V,W))_i = p(U,V,(W)_i),$$

i.e. the i-th attribute in the sequence p(U,V,W) is the result of applying p to U, V and the i-th attribute of W. The critical properties of p are summarized in the lemma below.

Lemma 2: Assume $(U \subseteq V)$, $(V \rightarrow B) \in \Sigma$.

(a) If B appears as the k-th attribute in V, then $p(U,V,B) \equiv (U)_k$.

(b) If $A = p(U,V,B)$, then $(UA \subseteq VB) \in \Sigma$.

(c) If $(U \subseteq V)$ follows from $(W \subseteq Z) \in \Sigma$ by permutation, projection and redundancy (rule I2), then $p(W,Z,B) \equiv p(U,V,B)$.

(d) If $(U \subseteq Z)$, $(Z \subseteq V) \in \Sigma$, then $p(U,V,B) \equiv p(U,Z,p(Z,V,B))$.

Proof: Properties (a) and (b) are easy consequences of the definition and Lemma 1. To see that (c) is true, let $A = p(U,V,B)$ and let $C = p(W,Z,B)$. By property (b), we have

$$UA \subseteq VB \quad \text{and} \quad WC \subseteq ZB$$

in Σ. Since $(U \subseteq V)$ is a projection and permutation of $(W \subseteq Z)$, the inclusion $(UC \subseteq VB)$ must be a projection and permutation of $(WC \subseteq ZB)$. Therefore $(UC \subseteq VB) \in \Sigma$. Thus $p(U,V,B) = A \equiv C$ by (a).

The remaining case is (d). Let $A = p(U,V,B)$, $C = p(Z,V,B)$ and $D = p(U,Z,C)$. It must be shown that $D \equiv A$. Since $UD \subseteq ZC$ and $ZC \subseteq VB$, we have $UD \subseteq VB$. Therefore, from $UA \subseteq VB$ and $UD \subseteq VB$, we conclude $D \equiv A$. ∎

Constructing the Counterexample Relation

At each stage in the construction, we verify inductively that the following property holds of the relation produced at that stage:

(*) For any pair of tuples t_j, t_k, if $t_j[X] = t_k[Y]$ for any sequences of attributes X and Y, then there is some common ancestor $t_i \leq t_j,t_k$ and some sequence of attributes Z such that

$$t_i[Z] = t_j[X] = t_k[Y]$$

Furthermore, $(Z \subseteq X)$, $(Z \subseteq Y) \in \Sigma$ and, for any attribute A, if $(X \rightarrow A) \in \Sigma$ then

$$t_j[A] = t_i[p(Z,X,A)]$$

and similarly if $(Y \rightarrow B) \in \Sigma$ then

$$t_k[B] = t_i[p(Z,Y,B)].$$

We begin the construction by choosing two tuples t_0 and t_1 to ensure that the functional dependency $(X_0 \rightarrow Y_0)$ fails. Let X_0^+ consists of all attributes functionally determined by X_0, i.e.

$$X_0^+ = \{A \mid (X_0 \rightarrow A) \in \Sigma\}.$$

The first tuple t_0 is chosen to have any arbitrary, distinct elements of S as entries, subject to the restriction that $t_0[A] = t_0[B]$ iff $A \equiv B$. For each attribute $A \in X_0^+$, let $t_1[A] = t_0[A]$. For each $A \notin X_0^+$, let $t_1[A]$ be some new element of S not appearing in t_0. Again, the entries must satisfy the equality constraint: $t_1[A] = t_1[B]$ iff $A \equiv B$. To avoid special cases in the remainder of the proof, we say that t_0 is the predecessor of t_1. Hence $t_0 \leq t_0$ and $t_0 \leq t_1$.

It is easy to see that the relation $r_1 = \{t_0, t_1\}$ satisfies all functional dependencies in Σ, as follows. Suppose that $t_0[X] = t_1[Y]$. By construction, $t_0[A] = t_1[B]$ iff $A \equiv B$ and $A, B \in X_0^+$. Therefore Y must be obtained from X by substitution of equivalent attributes and each attribute in X must appear in X_0. Thus, for any $(X \rightarrow B) \in \Sigma$, we have $B \in X_0^+$ and hence $t_0[B] = t_1[B]$. This also demonstrates (*) for the first stage of the construction.

We now add more tuples, producing a sequence of relations $r_1 \subseteq r_2 \subseteq \ldots$ such that the relation $r = \bigcup_k r_k$ satisfies all inclusion dependencies in Σ and such that (*) holds in each r_k. Let $(U_1 \subseteq V_1)$, $(U_2 \subseteq V_2)$, ... be an enumeration of inclusion dependencies from Σ such that for every $(U \subseteq V) \in \Sigma$, there are infinitely many i such that $(U \subseteq V)$ is a projection and permutation of $(U_i \subseteq V_i)$. The tuple t_k produced at stage k is chosen by looking at $(U_k \subseteq V_k)$.

Let r_k be the result of the k-th stage. If r_k satisfies $(U_k \subseteq V_k)$, then let r_{k+1} be r_k. Otherwise, let t_i be the tuple with lowest index such that $t_i[U_k]$ is not in $r_k[V_k]$. The tuple t_i will be the predecessor of t_{k+1}. The entries of t_{k+1} are chosen as follows. For each attribute B such that $(V_k \rightarrow B) \in \Sigma$, let

$$t_{k+1}[B] = t_i[p(U_k, V_k, B)].$$

For each attribute C not functionally determined by V_k, let $t_{k+1}[C]$ be some new element of S not appearing in r_k. Choose all such $t_{k+1}[C]$ so that $t_{k+1}[C] = t_{k+1}[D]$ iff $C \equiv D$. Note that since $p(U_k, V_k, V_k) \equiv U_k$, we have $t_{k+1}[V_k] = t_i[U_k]$.

We now verify (*) for r_{k+1}. Since (*) holds for r_k, we need only consider the effect of adding t_{k+1}. Suppose that there is some tuple t_j in r_k with $t_j[X] = t_{k+1}[Y]$ for some sequences of attributes X and Y. Then by the choice of symbols in t_{k+1}, all the entries in $t_{k+1}[Y]$ must have been entries in t_i. Hence $(V_k \rightarrow Y) \in \Sigma$. Let $W = p(U_k, V_k, Y)$. For each attribute $(W)_m$ of the sequence of attributes W, the construction ensures that

$$t_i[(W)_m] = t_{k+1}[(Y)_m].$$

By Lemma 2, each dependency $U_k(W)_m \subseteq V_k(Y)_m$ is in Σ. Since each $(V_k \rightarrow (Y)_m) \in \Sigma$, it follows from FI2 that $(U_kW \subseteq V_kY) \in \Sigma$. Thus $(W \subseteq Y) \in \Sigma$ by permutation and projection. Since $t_i[(W)_m] = t_{k+1}[(Y)_m]$ for all m, $t_i[W] = t_{k+1}[Y]$. We now have $t_{k+1}[Y] = t_i[W] = t_j[X]$ and $(W \subseteq Y) \in \Sigma$.

Since $t_i, t_j \in M_k$, it follows from the induction hypothesis (*) for M_k that there is some $t_n \leq t_i,t_j$ such that $t_n[Z] = t_i[W] = t_j[X]$ for some sequence of attributes Z. Furthermore, $(Z \subseteq W)$ and $(Z \subseteq X) \in \Sigma$. By transitivity of equality, $t_n[Z] = t_{k+1}[Y]$ and by transitivity of inclusion dependencies, $(Z \subseteq Y) \in \Sigma$. Thus

$$t_n[Z] = t_j[X] = t_{k+1}[Y]$$

and

$$(Z \subseteq X), (Z \subseteq Y) \in \Sigma.$$

To finish the proof of (*), it must be shown that if $(X \to A) \in \Sigma$, then $t_j[A] = t_n[p(Z,X,A)]$ and similarly $(Y \to B) \in \Sigma$ implies $t_{k+1}[B] = t_n[p(Z,Y,B)]$. The first case, if $(X \to A)$, is a trivial consequence of the induction hypothesis. Now suppose $(Y \to B) \in \Sigma$. Let $C = p(W,Y,B)$. Then $(WC \subseteq YB) \in \Sigma$ and, by FI1, $(W \to C) \in \Sigma$. Thus $t_i[C] = t_n[p(Z,W,C)]$. Let $D = p(Z,Y,B)$. By Lemma 2, $D \equiv p(Z,W,C)$. It remains to show that $t_{k+1}[B] = t_n[D]$. First note that since $(U_kW \subseteq V_kY)$ extends $(W \subseteq Y)$, and both $(Y \to B)$, $(V_k \to B) \in \Sigma$, we have $p(U_k,V_k,B) \equiv p(U_kW, V_kY, B) \equiv C$. Therefore $t_{k+1}[B] = t_i[C]$. Recall that $t_i[C] = t_n[p(Z,W,C)]$. But since $D \equiv p(Z,W,C)$, it follows that $t_i[C] = t_n[D]$. Therefore

$$t_{k+1}[B] = t_i[C] = t_n[D].$$

This demonstrates (*) for M_{k+1}.

Now consider the relation $r = \cup_k r_k$. To see that r satisfies all functional dependencies in Σ, let $X \to Y \in \Sigma$ and suppose that there are two tuples t_j and t_k in r with $t_j[X] = t_k[X]$. By (*), there is some $t_i \leq t_j, t_k$ such that

$$t_i[W] = t_j[X] = t_k[X] \quad \text{and} \quad (W \subseteq X) \in \Sigma.$$

Furthermore, for all $m \leq |Y|$,

$$t_j[(Y)_m] = t_i[p(W,X,(Y)_m)] = t_k[(Y)_m].$$

Thus $t_j[Y] = t_k[Y]$ and $(X \to Y)$ holds. All functional dependencies in Σ are satisfied by r, but by choice of t_0 and t_1 the functional dependency $X_0 \to Y_0$ is not.

In addition, the relation r satisfies $X \subseteq Y$ iff $X \subseteq Y \in \Sigma$. This is demonstrated as follows. It is clear from the construction that if $X \subseteq Y \in \Sigma$, then for any t_i there is some r_k with $t_i[X] \in r_k[Y]$. Thus r satisfies all $X \subseteq Y$ in Σ. For the converse, assume $(X \subseteq Y) \notin \Sigma$. We show that $t_0[X] \notin r[Y]$ using property (*). Suppose that, on the contrary, there is some tuple t_k in r_k with $t_0[X] = t_k[Y]$. Then by (*) there is some $t_i \leq t_0, t_k$ with $t_i[Z] = t_0[X] = t_k[Y]$ and $(Z \subseteq Y), (Z \subseteq X) \in \Sigma$. But the only tuple t_i with $t_i \leq t_0$ is $t_i = t_0$. Also, by construction of t_0, we have $t_0[Z] = t_0[X]$ iff Z may be obtained from X by substituting equivalent attributes. Therefore, by substitutivity of equivalents and $Z \subseteq Y \in \Sigma$ we conclude $X \subseteq Y \in \Sigma$. Since this is a contradiction, it follows that $t_0[X] \neq t_k[Y]$. Thus r satisfies $X \subseteq Y$ iff $X \subseteq Y \in \Sigma$. In particular, r does not satisfy $X_0 \subseteq Z_0$ since this dependency does not appear in Σ. This finishes the proof of Theorem 1.

5. Undecidability

A simple translation of equations into dependencies shows that both the finite and general implication problems are undecidable. We know that the valid general implications are recursively enumerable since the dependencies are first-order formulas. Since a simple enumeration of finite databases will uncover all *invalid* finite implications (from finite sets of hypotheses), the *valid* finite implications for functional and inclusion dependencies form the complement of a recursively enumerable set. The reduction described below will show that both problems are as hard as any in their respective classes (cf. [18]).

Intuitively, the idea of the reduction is to use functional dependencies and inclusion dependencies to force the pairs of columns of a relation to contain functions (i.e. graphs of functions) from some arbitrary set to itself. Since any monoid (semigroup with unit; cf. [18]) is isomorphic to a monoid of functions from a set to itself, the relations satisfying this set of dependencies correspond to arbitrary monoids. Using inclusion dependencies, we can then express equations between compositions of functions. This translation of equations to dependencies provides reductions from the word problems for monoids and finite monoids to the general and finite implication problems, respectively. Although the reduction is not complicated, it is given in some detail to make the presentation more readable and self-contained.

The translations from monoid equations to dependencies is simplified by adopting a slightly peculiar syntax for equations. A *signature* s is a pair $\langle A,X\rangle$ where A is an attribute and X is a set of attributes with $A \in X$. A *composition equation* over a signature $s = \langle A,X\rangle$ is an equation of the form

$$AB = AC \circ AD$$

where $B, C, D \in X$. The pairs of symbols AB, with $B \in X$, are called the *terms* of s. A *monoid interpretation* for a signature s is a monoid M together with a mapping ρ from terms of s to elements of the monoid. We assume that $\rho(AA)$ is the unit of the monoid. A monoid interpretation $\langle M, \rho\rangle$ satisfies an equation

$$AB = AC \circ AD$$

if $\rho(AB)$ is the product of $\rho(AC)$ and $\rho(AD)$ in M. If $T \cup \tau$ is a set of composition equations, then $T \models \tau$ means that every monoid interpretation satisfying T also satisfies τ. We use $\models_{\text{finite}}$ for implication over finite monoids. Since $\rho(AA)$ is a unit, we can write $AB = AC$ by writing $AB = AC \circ AA$.

The word problem for monoids is well-known to be undecidable [20] (see also [18]). A convenient version of the word problem is the following implication problem:

> Given a finite set $T \cup \{\tau\}$ of composition equations, determine whether τ holds in every monoid satisfying T.

In the corresponding finite version, we ask instead whether τ holds in every finite monoid satisfying T. The finite implication problem (word problem for finite monoids) is proved undecidable in [13] (see also [14]).

Composition equations can be interpreted over any relation if the appropriate attributes of the relation contain functions which generate a monoid. Fortunately, at least as far as the proof goes, this is a property which can be described using functional and inclusion dependencies. If $s=\langle A,X\rangle$ is a signature, then let Σ_s denote the set of dependencies

$$\Sigma_s = \{A \rightarrow B \mid B \in X\} \cup \{B \subseteq A \mid B \in X\}.$$

A *relational interpretation* for a signature $s = \langle A,X\rangle$ is a relation r satisfying Σ_s. Note that if r is a relational interpretation for $s = \langle A,X\rangle$ and $B \in X$, then the set of ordered pairs $r[AB]$ is a function (in the set-theoretic sense, i.e. the graph of a function) from $r[A]$ to $r[A]$. Furthermore, $r[AA]$ is the identity function on $r[A]$.

If τ is a composition equation $AB = AC \circ AD$ over some signature s and r is a relational interpretation for s, then r *satisfies* τ if the function r[AB] is the composition of the functions r[AC] and r[AD], i.e.

$$r[AB] = \{ \langle a,b \rangle \mid \exists c \text{ with } \langle a,c \rangle \in r[AC] \text{ and } \langle c,b \rangle \in r[AD] \}.$$

We can express composition equations as inclusion dependencies, as shown in the following lemma.

Lemma 3: Let τ be a composition equation $AB = AC \circ AD$ over signature s and let r be a relational interpretation for s. Then r satisfies τ iff r satisfies $CB \subseteq AD$.

Proof: First suppose that r is a relational interpretation which satisfies $AB = AC \circ AD$. Let b, c and d denote the functions r[AB], r[AC] and r[AD] respectively. Then for any tuple $t \in r$, we have

$$t[B] = b(t[A]) = d(c(t[A])) = d(t[C]).$$

Since r is a relational interpretation, we know $r[C] \subseteq r[A]$ and so there is some tuple $t_1 \in r$ with $t_1[A] = t[C]$. Therefore

$$t[CB] = \langle t[C], d(t[C]) \rangle = \langle t_1[A], d(t_1[A]) \rangle = t_1[AD].$$

This shows that r satisfies $CB \subseteq AD$.

Now assume that r satisfies $CB \subseteq AD$. For any tuple $t \in r$, there is a tuple $t_1 \in r$ with $t[CB] = t_1[AD]$. Therefore, for functions b, c and d as above, we have

$$t_1[A] = c(t[A]) \quad \text{and} \quad b(t[A]) = d(t_1[A]).$$

By substituting $c(t[A])$ for $t_1[A]$, we obtain

$$b(t[A]) = d(t_1[A]) = d(c(t[A])).$$

Since this holds for all t[A], i.e. all elements of the domain of b, c and d, we can conclude that $b = c \circ d$. Thus r satisfies $AB = AC \circ AD$. ∎

If τ is the composition equation $AB = AC \circ AD$, then we call $CB \subseteq AD$ the *dependency translation* of τ and write $(CB \subseteq AD) = \mathit{Trans}(\tau)$. If T is a set of composition equations, then *Trans*(T) is the set of dependency translations of equations from T.

Although the set of functions given by a relational interpretation r need not be closed under composition, any relational interpretation can be expanded to a monoid interpretation. This is the content of the lemma below.

Lemma 4: Let r be a relational interpretation for s. There is a monoid interpretation $\langle M, \rho \rangle$ for s which satisfies precisely the same composition equations over s as r. Furthermore, if r is a finite relation then M is a finite monoid.

Proof: Define the set of generators from the interpretation r for S = ⟨A,X⟩ by

$$GEN_{r,s} = \{ r[AB] \mid B \in X \}$$

and let M be the smallest set of functions from r[A] to r[A] containing $GEN_{r,s}$ and closed under composition. Define ρ by $\rho(AB) = r[AB]$. It is clear that $\langle M,\rho\rangle$ satisfies the same equations over s as the relational interpretation r. In addition, if r is a finite relation then r[A] is finite and so there are only finitely many equations which can be in M.∎

Lemma 4 has the following converse.

Lemma 5: If $\langle M,\rho\rangle$ is a monoid interpretation for s = ⟨A,X⟩ then there is some relational interpretation r which satisfies precisely the same equations over s as $\langle M, \rho\rangle$. Furthermore, if M is a finite monoid then r is a finite relation.

Proof: The relation r will be constructed using a tuple for each element of M and using elements of M as entries. For each element $m \in M$, let t_m be any tuple such that

$$t_m[B] = (\rho(AB))(m)$$

for all $B \in X$. In particular, $t_m[A] = m$. The remainder of the proof follows the usual proof of Cayley's Theorem (cf. [15]) for monoids. Since there are as many tuples in r as elements of M, the relation r will be finite whenever M is. ∎

The main results of this Section follow easily from the equivalence between monoid implication and relational database implication stated below.

Lemma 6: Let $T \cup \{\tau\}$ be a set of composition equations over some signature s = ⟨A,X⟩. For $\Sigma = \mathit{Trans}(T) \cup \Sigma_s$ and $\sigma = \mathit{Trans}(\sigma)$ we have the following equivalences.

(i) $T \models \tau$ iff $\Sigma \models \sigma$

(ii) $T \models_{\text{finite}} \tau$ iff $\Sigma \models_{\text{finite}} \sigma$

Proof: We first show that if Σ does not imply σ, then T does not imply τ. If Σ does not imply σ, then there is a relation r which satisfies Σ but not σ. Since r satisfies $\Sigma_s \subseteq \Sigma$, r is a relational interpretation for s. Therefore, by Lemma 3, r satisfies T but not τ. Furthermore, by Lemma 4, there is a monoid interpretation $\langle M,\rho\rangle$ which satisfies precisely the same equations over s as r. Thus $\langle M,\rho\rangle$ satisfies T but not τ. Note also that if r is finite then so is M. This proves half of each equivalence.

For the converses, suppose that $\langle M,\rho\rangle$ is a monoid interpretation for s which satisfies T but not τ. Then by Lemma 5 there is a relational interpretation r which satisfies T but not τ. But then by Lemma 3, r satisfies Σ but not σ. Recall that if M is finite then so is r. This concludes the proof of the lemma. ∎

We now have

> **Theorem 2:** The implication and finite implication problems for functional dependencies and inclusion dependencies are recursively unsolvable.

as a simple consequence of the undecidability results of [20] and [13].

6. Conclusion

This paper presents a complete axiom system for functional dependencies and inclusion dependencies. The system stands in contrast to the possibility suggested in [5] that no such system exists. Essentially, the difficulties discussed in [5] are surmounted using an inference rule similar to existential instantiation in a natural deduction system. A rule which allows new attribute names to be introduced into deductions simplifies reasoning about functional and inclusion dependencies.

Both the finite implication and general implication problems are shown to be undecidable. The proof uses the simple observation that functional dependencies force projections of a relation to be functions and inclusion dependencies can express equality between compositions of functions. This reduces the word problems for monoids and finite monoids to the general implication and finite implication problems for dependencies. Since there is no complete axiomatization for finite monoids, there is no complete axiomatization for finite implication. It is interesting to note that when relations are interpreted as monoids, introducing new attribute names corresponds to naming products in a monoid.

Although the implication and finite implication problems are both undecidable, there are restricted versions of these problems with polynomial-time decision procedures [17]. For example, as suggested in [5], one may consider functional dependencies together with simple inclusion dependencies of the form $A \subseteq B$, where A and B are both single attributes. These restricted inclusion dependencies are called *unary* inclusion dependencies. In [17] it is shown that implication for functional dependencies and unary inclusion dependencies is decidable in polynomial time. A polynomial-time decision procedure for finite implication of functional dependencies and *unary* inclusion dependencies is also given in [17], along with a complete axiom system for finite implication.

The translation presented in Section 5 of monoid equations into dependencies uses only simple *binary* inclusion dependencies of the form AB $\subseteq$ CD, where A, B, C and D are single attributes. Thus the results of [17] cannot be extended even to *binary* inclusion dependencies.

Thanks to Christos Papadimitriou, Albert Meyer and, in particular, Paris Kanellakis for many helpful discussions.

7. References

1. Armstrong, W.W. Dependency Structures of Database Relationships. Proc. IFIP '74, 1974, pp. 580-583.

2. Beeri, C. and P.A. Bernstein. Computational Problems Related to the Design of Normal Form Relational Schemes. *ACM Trans. on Database Systems 4*, 1 (March 1979). pp 30-59

3. Beeri, C., R. Fagin and J.H. Howard. A Complete Axiomatization for Functional and Multivalued Dependencies in Database Relations. Proc. 1977 ACM SIGMOD Conference, 1977. pp 47-61

4. Beeri, C. and H.F. Korth. Compatible Attributes in a Universal Relation. Proc. 1st ACM Conf. on Principles of Database Systems, 1982.

5. Casanova, M.A., R. Fagin and C.H. Papadimitriou. Inclusion Dependencies and Their Interaction with Functional Dependencies. Proc. 1st ACM Conf. on Principles of Database Systems, 1982, pp. 171-176. Also appears as IBM Research Report RJ3380 (40416), 1982.

6. Chen, P. The Entity-Relationship Model -- Toward a Unified View of Data. *ACM Trans. on Database Systems 1*, 1 (March 1976). pp 9-36

7. Codd, E.F. Relational Completeness of Database Sublanguages. In R. Rustin, Ed., *Data Base Systems*, Prentice-Hall, New Jersey, 1972.

8. Codd, E.F. Extending the Database Relational Model to Capture More Meaning. *ACM Trans. on Database Systems 4*, 4 (December 1980). pp 397-434

9. Chandra, A. and M. Vardi. Personal Communication (Vardi). January 1983.

10. Fagin, R. A Normal Form for Relational Databases that is Based on Domains and Keys. *ACM Trans. on Database Systems 6*, 3 (September 1981). pp 387-415

11. Fagin, R. Horn Clauses and Database Dependencies. Proc. ACM Symp. on Theory of Computing, 1980, pp. 123-143. To appear in (JACM).

12. Garey, M.R. and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W.H. Freeman and Co., 1979.

13. Gurevich, Y. The Word Problem for Certain Classes of Semigroups. *Algebra and Logic 5*, 5 (1966). pp 25-35.

14. Gurevich, Y. and H.R. Lewis. The Word Problem for Cancellation Semigroups with Zero. Tech. Rep. TR-08-82, Harvard, 1982.

15. Herstein, I.N. *Topics in Algebra.* Xerox, 1975.

16. Johnson, D.S. and A. Klug. Testing Containment of Conjunctive Queries Under Functional and Inclusion Dependencies. Proc. 2nd ACM Conf. on Principles of Database Systems, 1982, pp. 164-169.

17. Kanellakis, P.C., S.S. Cosmadakis, M.Y. Vardi. Unary Inclusion Dependencies Have Polynomial Time Inference Problems. To Appear 15-th ACM Symposium on Theory of Computing, April, 1983.

18. Machtey, M. and P. Young. *An Introduction to the General Theory of Algorithms.* North Holland, 1978.

19. Mitchell, J.C. Inference Rules for Functional and Inclusion Dependencies. Proc. 2nd ACM Conf. on Principles of Database Systems, March, 1983.

20. Post, E.L. Recursive Unsolvability of a Problem of Thue. *Journal of symbolic Logic 12* (1947). pp 1-11.

21. Vardi, M.Y. The Implication and Finite Implication Problems for Typed Template Dependencies. Tech. Rep. STAN-CS-82-912, Stanford University, 1982.

22. Yannakakis, M. and C.H. Papadimitriou. Algebraic Dependencies. Proc. 21-st IEEE Symp. on Found. of Comp. Sci., 1980, pp. 328-332.

Table of Contents