

## The Implication Problem for Functional and Inclusion Dependencies

JOHN C. MITCHELL\*

*Laboratory for Computer Science, Massachusetts Institute of Technology,  
Cambridge, Massachusetts 02139*

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  $\sigma$  holds in all databases satisfying  $\Sigma$  or all finite databases satisfying  $\Sigma$ . Contrary to the possibility suggested in Casanova, Fagin, and Papadimitriou ("Proceedings, 1st ACM Conf. on Principles of Database System," pp.171-176, 1982), 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.

### 1. INTRODUCTION

Functional dependencies have been discussed extensively in the literature on relational databases (e.g., Armstrong, 1974, Beeri and Bernstein, 1979, Beeri, Fagin, and Howard, 1977, Casanova, Fagin, and Papadimitriou, 1982). 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  $\text{EMPLOYEE} \rightarrow \text{MANAGER}$ . For functional dependencies, finite and general implication coincide. Implication for functional dependencies has a well-known axiomatization (Armstrong, 1974) and an efficient decision procedure (Beeri and Bernstein, 1979). Inference rules and decision procedures have also been developed for functional dependencies in combination with various other dependencies (e.g., Beeri *et al.*, 1977, Yannakakis and Papadimitriou, 1982).

Although inclusion dependencies are common in database practice (Beeri and Korth, 1982, Chen, 1976, Codd, 1980, Fagin, 1981), the theoretical properties of inclusion dependencies have received relatively little attention

\* The author was supported by a fellowship from the National Science Foundation.

until quite recently. An inclusion dependency arises in the EMPLOYEE and MANAGER database. In a typical corporation, every MANAGER is also an EMPLOYEE. Hence the set of employees in any office database will include the set of managers in the database. This inclusion dependency is written  $\text{MANAGER} \subseteq \text{EMPLOYEE}$ . As for functional dependencies, implication and finite implication coincide for inclusion dependencies. Recent theoretical papers on inclusion and functional dependencies include Casanova *et al.* (1982) and Johnson and Klug (1982). In particular, Casanova *et al.* (1982) describes the interaction between functional and inclusion dependencies and discusses previous work by other authors. In Casanova *et al.* (1982), 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. Garey and Johnson, 1979).

General implication and finite implication differ when functional dependencies and inclusion dependencies are considered together (Casanova *et al.*, 1982). Since we will be concerned most often 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 Casanova *et al.* (1982). A dependency  $\sigma$  follows from a set of dependencies  $\Sigma$  by *k-ary implication* if there is some subset of *k* dependencies from  $\Sigma$  that implies  $\sigma$ . In Casanova *et al.* (1982), 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 for general, but not finite, implication. The rules differ from those considered by Casanova *et al.* (1982) in two respects. A relatively 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. A more important difference is that 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 Casanova *et al.* (1982). The inference rules of the system are all 3-ary since each rule yields a single new conse-

quent by inspection of a most three antecedents. However, *attribute introduction* is not sound in the usual sense. The inference system is also “universe unbounded” (cf. Vardi, 1982) 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 one 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,  $MGR \subseteq EMP$ . 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 may add a new attribute MSAL to the database and write  $MGR \rightarrow MSAL$ . The entries in the new column MSAL, with

$$MGR, MSAL \subseteq EMP, SAL$$

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

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

The attribute introduction rule simplifies reasoning about functional and inclusion dependencies by introducing new attributes like MSAL. The entries of the new attributes may be thought of as computed or derived from the entries of the original database. Intuitively, the main use of new attributes in proofs 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 view leads 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

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

to mean that the salary entry in any tuple (or “row”) of the database is the

result of applying the  $\{\text{EMP} \rightarrow \text{SAL}\}$  function to the employee entry in that tuple. This function  $\{\text{EMP} \rightarrow \text{SAL}\}$  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  $\{\text{EMP} \rightarrow \text{SAL}\}$  since  $\text{MGR} \subseteq \text{EMP}$ . Instead of using new attribute names like MSAL in deductions, we could use applicative expressions built from original attribute names.

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, functional and inclusion dependencies are intractable because these dependencies may make statements about compositions of functions. The implication problem for monoids (word problem) can be reduced to the general implication problem for functional and inclusion dependencies by translating equations between compositions 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 (Gurevich, 1966, Gurevich and Lewis, 1982), there is no complete, recursively enumerable axiomatization for finite implication of inclusion dependencies and functional dependencies.<sup>1</sup>

## 2. DATABASES AND DEPENDENCIES

A relational database consists of a set of relations. To keep the notation simple, all inference rules presented in this paper are written for functional and inclusion dependencies which mention only one relation. All the rules can be rewritten to apply to arbitrary database schemes; see Casanova *et al.* (1982) for examples of dependencies involving more than one relation. The

<sup>1</sup> Both undecidability results have also been obtained independently by Chandra and Vardi using different methods of proof (Chandra and Vardi, 1983).

completeness proof in Section 4 is also easily extended to arbitrary databases of nonempty relations.

Formally, a relation name  $R$  has associated attributes  $R[1], R[2], \dots$ . 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. A tuple  $t \in r$  is a sequence of *entries*  $\langle a_1, a_2, \dots \rangle$ . We write  $t[i]$  to denote the  $i$ th entry of  $t$ . If  $X$  is a finite sequence of attributes  $\langle X_1, X_2, \dots \rangle$ , then  $t[X]$  denotes the sequence of entries  $\langle t[X_1], t[X_2], \dots \rangle$  and  $|X|$  denotes the length of  $X$ . Note that an 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.

Following common convention, capital letters from the beginning of the alphabet  $A, B, C, \dots$  will be used to denote single attributes while capital letters from the end of the alphabet  $U, V, W, X, \dots$  will denote nonempty sequences of attributes. Lowercase  $s$  and  $t$ , possibly with subscripts, will denote tuples and  $r$  a relation.

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  $B \neq A$ , we have  $f(t)[B] = t[B]$ . Intuitively, an  $A$ -variant of a relation  $r$  is another relation which differs from  $r$  only on attribute  $A$ .

A *functional dependency* is an assertion of the form  $X \rightarrow Y$ , where  $X$  and  $Y$  are nonempty sequences of attributes. A relation  $r$  satisfies  $X \rightarrow Y$  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  $X \subseteq Y$ . A relation  $r$  satisfies  $X \subseteq Y$  if  $r[X] \subseteq r[Y]$ . The expression

$$\Sigma \models \sigma$$

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

### 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.<sup>2</sup> The salient new rule of the system is the attribute introduction rule,

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

<sup>2</sup>Two combined rules, listed as (FI1) and (FI2), were discovered independently by the author and by Casanova *et al.* (1982). The functional dependency rules (F1)–(F3) are essentially from Armstrong (1974) and the inclusion dependency rules (I1)–(I3) from Casanova *et al.* (1982). The functional dependency rules of Armstrong (1974) produce dependencies between sets of attributes, rather than ordered sequences of attributes.

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 a definition of proof which ensures that  $A$  is “new,” all proofs of the system will be sound. Proofs will be defined after the axioms and rules are presented.

### *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  $A_1, \dots, A_n \subseteq B_1, \dots, B_n$  derive  $A_{i_1}, \dots, A_{i_k} \subseteq B_{i_1}, \dots, B_{i_k}$ , 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  $\sigma$  derive  $\tau$ , where  $\tau$  is obtained from  $\sigma$  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  $\Sigma$  denote a set of functional dependencies and inclusion dependencies. A *proof from  $\Sigma$*  is a sequence of dependencies  $\langle \sigma_1, \dots, \sigma_n \rangle$  such that:

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

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

An inclusion or functional dependency  $\sigma$  is *provable from*  $\Sigma$ , written  $\Sigma \vdash \sigma$  if there is some proof  $\langle \sigma_1, \dots, \sigma_n \rangle$  from  $\Sigma$  with  $\sigma = \sigma_n$  and such that no attributes in  $\sigma$  are new in  $\sigma_1, \dots, \sigma_n$ .

**THEOREM 1 (Completeness).** *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  $\langle \sigma_1, \dots, \sigma_n \rangle$  from  $\Sigma$  shows that if a relation  $r$  satisfies  $\Sigma$ , then there is a relation  $r'$  which differs from  $r$  only on new attributes of the proof and which satisfies each  $\sigma_i$ . It follows that the inference system is sound. The only complicated cases of the induction are the cases for (FI3) and (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  $\langle v_1, \dots, v_k \rangle \in r[V]$ , define  $g(v_1, \dots, v_k)$  by

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

Since  $r$  satisfies  $V \rightarrow B$ , this condition defines a 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

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

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 Mitchell (1983), 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 Casanova *et al.* (1982), 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  $\Sigma$  consisting only of repeating dependencies and inclusion dependencies without repeated attributes. A simple modification to the proof presented in Casanova *et al.* (1982) extends their “no  $k$ -ary axiomatization” theorem to the slightly more powerful dependencies with repeated attributes.

**THEOREM** (Casanova *et al.*, 1982). *For every  $k$ , there is a set  $\Sigma$  of inclusion and functional dependencies such that all consequences of every subset of  $\Sigma$  of size  $k$  are included in  $\Sigma$ , yet  $\Sigma$  is not closed under implication.*

The inclusion dependency rules (I1)–(I3) are taken from Casanova *et al.* (1982) and are shown there to be complete for inclusion dependencies without repeated attributes. Specifically, if  $\Sigma$  is a set of inclusion dependencies without repeated attributes and  $\sigma$  is another such dependency, then  $\Sigma$  implies  $\sigma$  iff  $\sigma$  is provable from  $\Sigma$  by (I1)–(I3).

It may be shown that (I1)–(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  $\Sigma$  is a set of inclusion dependencies without repeated attributes and  $\Sigma$  implies the inclusion dependency  $\sigma$ , then  $\sigma$  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 Casanova *et al.* (1982) 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  $\Sigma$  be the following set of hypotheses:



- (h1)  $C \rightarrow D$ ,
- (h2)  $AB \subseteq CD$ ,
- (h3)  $BA \subseteq CD$ ,
- (h4)  $B \subseteq A$ ,

and let  $\sigma$  be  $AB \subseteq BA$ . The reader may verify that  $\sigma$  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  $\Sigma$  implies  $\sigma$ . Consider any tuple  $t_1$  in any relation  $r$  satisfying  $\Sigma$ . 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  $t_3[D] = t_4[D]$ . Thus  $x = a$ , which proves that  $\sigma$  follows from  $\Sigma$ . We can prove  $\sigma$  from  $\Sigma$  using the inference rules as follows:

- (1)  $A \rightarrow B$  from (h1) and (h2) by (FI1).
- (2)  $BE \subseteq AB$  by (FI3) from (h4) and (1); note that  $E$  does not appear in  $\Sigma$  or previously in the proof.
- (3)  $BE \subseteq CD$  from (2) and (h2) by (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).

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. Let  $\Sigma_0$  be a set of dependencies and  $\sigma$  a dependency that is not provable from  $\Sigma_0$  by the attribute introduction rules. Theorem 1 is proved by constructing a relation that satisfies  $\Sigma_0$  but not  $\sigma$ . 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:  $\sigma$  may be an inclusion dependency or  $\sigma$  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  $\sigma$  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  $\sigma$  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  $\Sigma_0$  but not  $\sigma$ , then there is also a relation that satisfies  $\Sigma_0$  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  $\Sigma_0$ .

A set of dependencies  $\Sigma$  is *deductively closed* if  $\Sigma$  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  $\Sigma_0$  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  $\Sigma_1$  has the same consequences as  $\Sigma_0$  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  $\Sigma_1$  will not be attributes which appear in  $X_0, Y_0$  or  $Z_0$ . Let  $\Sigma \supseteq \Sigma_1$  be deductively closed with neither  $X_0 \rightarrow Y_0$  nor  $X_0 \subseteq Z_0$  an element of  $\Sigma$ .<sup>3</sup> Theorem 1 is proved by constructing a relation that satisfies  $\Sigma$  but does not satisfy either  $X_0 \rightarrow Y_0$  or  $X_0 \subseteq Z_0$ .

An outline of the construction will make the proof easier to follow. 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  $r_k$  produced so far to “help” satisfy some inclusion dependency  $U_k \subseteq V_k$  in  $\Sigma$ . This is done in such a way that all functional dependencies in  $\Sigma$  hold at each stage. Furthermore, no inclusion dependency which is not in  $\Sigma$  will be satisfied inadvertently. If the relation  $r_k$  produced at stage  $k$  does not satisfy  $U_k \subseteq V_k$ , then we pick a tuple  $t_i$  from  $r_k$  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  $r_k \cup \{t_{k+1}\}$ . 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

<sup>3</sup> We can construct a deductively closed set containing  $\Sigma_1$  in stages by adding dependencies used in proofs, including those with new attributes, to  $\Sigma_1$ . We begin with  $\Sigma_1$  at stage 1 and fix an enumeration  $\Pi_2, \Pi_3, \Pi_4, \dots$ , of all finite sequences of dependencies such that every sequence appears infinitely often in the enumeration. At stage  $i + 1$ , we either add all dependencies in  $\Pi_i$  to the set  $\Sigma_i$  so far, if  $\Pi_{i+1}$  is a valid proof from  $\Sigma_i$ , or else discard  $\Pi_i$  and keep  $\Sigma_{i+1} = \Sigma_i$ . Let  $\Sigma$  be the union of the  $\Sigma_i$ . Then every consequence of  $\Sigma_0$  is in  $\Sigma$ , and every dependency in  $\Sigma$  containing only attributes that appear in  $\Sigma_0$  is a consequence of  $\Sigma_0$ .

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  $r = \bigcup_k r_k$  will be shown to satisfy precisely the inclusion dependencies in  $\Sigma$ . 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  $\Sigma$ , the relation  $r$  will not satisfy  $\sigma$ .

### *Attribute Equivalence and Pullback Function*

In the remainder of the proof, with  $\Sigma$  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  $\Sigma$  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  $U$ . Lemma 1 shows 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  $\Sigma$  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  $\Sigma$  since  $\Sigma$  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  $\Sigma$  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

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 (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  $\Sigma$ . 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  $\Sigma$ , 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  $r_1 \subseteq r_2 \subseteq \dots$ , such that the relation  $r = \bigcup_k r_k$  satisfies all inclusion dependencies in  $\Sigma$  and such that (\*) holds in each  $r_k$ . Let  $(U_1 \subseteq V_1), (U_2 \subseteq V_2), \dots$ , be an enumeration of inclusion dependencies from  $\Sigma$  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  $\Sigma$ . Since each  $(V_k \rightarrow (Y)_m) \in \Sigma$ , it follows from (FI2) that  $(U_k W \subseteq V_k Y) \in \Sigma$ . 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  $(X \rightarrow A) \in \Sigma$ , then  $t_j[A] = t_n[p(Z, X, A)]$  and similarly  $(Y \rightarrow B) \in \Sigma$  implies  $t_{k+1}[B] = t_n[p(Z, Y, B)]$ . The first case, if  $(X \rightarrow A)$ , is a trivial consequence of the induction hypothesis. Now suppose  $(Y \rightarrow B) \in \Sigma$ . Let  $C = p(W, Y, B)$ . Then  $(WC \subseteq YB) \in \Sigma$  and, by (FI1),  $(W \rightarrow C) \in \Sigma$ . 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  $(U_k W \subseteq V_k Y)$  extends  $(W \subseteq Y)$ , and both  $(Y \rightarrow B)$ ,  $(V_k \rightarrow B) \in \Sigma$ , we have  $p(U_k, V_k, B) \equiv p(U_k W, V_k Y, B) \equiv C$ . 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  $r_{k+1}$ .

Now consider the relation  $r = \bigcup_k r_k$ . To see that  $r$  satisfies all functional dependencies in  $\Sigma$ , let  $X \rightarrow Y \in \Sigma$  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  $(X \rightarrow Y)$  holds. All functional dependencies in  $\Sigma$  are satisfied by  $r$ , but by choice of  $t_0$  and  $t_1$  the functional dependency  $X_0 \rightarrow Y_0$  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  $\Sigma$ . 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  $t_k$  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  $\Sigma$ . 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. The valid general implications are recursively enumerable since 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 complete in their respective classes (cf. Machtey and Young, 1978).

Intuitively, the main idea behind the reduction is to use function 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. Machtey and Young, 1978) 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.

A few definitions are in order before choosing a convenient form of the word problem. Let  $\mathcal{V}$  be an infinite set of variables. Variables from  $\mathcal{V}$  will be used to write equations between compositions of functions. For every variable  $x \in \mathcal{V}$ , we pick an attribute  $B_x$ . If  $x$  and  $y$  are different variables, then  $B_x$  and  $B_y$  are assumed to be different. In addition, we need an attribute  $A$  that is different from each  $B_x$ .

A *composition equation* is an equation

$$x = y \circ z,$$

where  $x, y, z \in \mathcal{V}$ . A *functional interpretation*  $I$  for a set of variables  $U \subseteq \mathcal{V}$  is a set  $S$  together with a function  $f_x: S \rightarrow S$  for each  $x \in U$ . A functional interpretation is *finite* if  $S$  is finite. An interpretation  $I$  satisfies a set of equations  $E$  if

$$\forall s \in S. f_x(s) = f_z(f_y(s))$$

for all  $(x = y \circ z) \in E$ . The word problem for monoids is well known to be undecidable (Post, 1947) (see, also, Machtey and Young, 1978). A convenient version of the word problem is the following implication problem:

Given a finite set  $T \cup \{\tau\}$  of composition equations, determine whether  $\tau$  holds in every functional interpretation that satisfies  $T$ .

In the corresponding finite version, we ask instead whether  $\tau$  holds in every finite functional interpretation satisfying  $T$ . The finite implication problem (word problem for finite monoids) is proved undecidable in Gurevich (1966)(see, also, Gurevich and Lewis, 1982).

Composition equations can be interpreted over any relation if the appropriate attributes of the relation contain functions from some set to itself. This is a property which can be described using functional and inclusion dependencies. If  $U \subseteq \mathcal{V}$  is a set of variables, let  $\Sigma_U$  be the set of dependencies

$$\Sigma_U = \{A \rightarrow B_x | x \in U\} \cup \{B_x \subseteq A | x \in U\}.$$

A *relational interpretation* for a set of variables  $U$  is a relation  $r$  satisfying  $\Sigma_U$ . Note that if  $r$  is a relational interpretation for  $U$  and  $x \in U$ , then the set of ordered pairs  $r[AB_x]$  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  $r[A]$ . A relational interpretation  $r$  satisfies a set of composition equations  $E$  if  $r[AB_x]$  is the composition of  $r[AB_y]$  and  $r[AB_z]$  for every  $(x = y \circ z) \in E$ .

LEMMA 3. *Let  $T \cup \{\tau\}$  be a set of composition equations using variables from some subset  $U \subseteq \mathcal{V}$ . The following two conditions are equivalent.*

- (i) *Every functional interpretation (finite functional interpretation) for  $U$  that satisfies  $T$  also satisfies  $\tau$ .*
- (ii) *Every relational interpretation (finite relational interpretation) for  $U$  that satisfies  $T$  also satisfies  $\tau$ .*

*Proof.* Let  $I$  be a functional interpretation for  $U$  using functions from  $S$  to  $S$ . We construct a relational interpretation  $r$  that satisfies exactly the same composition equations as  $I$ . Let  $r[A] = S$  and, for each tuple, let



$t[B_x] = f_x(t[A])$  for all  $x \in U$ . Note that if  $I$  is a finite functional interpretation, then  $r$  is a finite relation. The details are straightforward.

Conversely, if  $r$  is a relational interpretation, we can define a functional interpretation satisfying precisely the same composition equations by letting  $S = r[A]$  and  $f_x = r[AB_x]$ . Again, the details are straightforward and  $I$  is finite if  $r$  is a finite relation. ■

We can express composition equations as inclusion dependencies, as shown in

LEMMA 4. *Let  $\tau$  be a composition equation  $x = y \circ z$  and let  $r$  be a relational interpretation for any set of variables containing  $x, y$  and  $z$ . Then  $r$  satisfies  $\tau$  iff  $r$  satisfies  $B_y B_x \subseteq AB_z$ .*

*Proof.* First suppose that  $r$  is a relational interpretation which satisfies  $x = y \circ z$ . For each  $v \in \mathcal{V}$ , let  $f_v$  denote the function  $r[AB_v]$ . Then for any tuple  $t \in r$ , we have

$$t[B_x] = f_x(t[A]) = f_z(f_y(t[A])) = f_z(t[B_y]).$$

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

$$t[B_y B_x] = \langle t[B_y], f_x(t[B_y]) \rangle = \langle t_1[A], f_x(t_1[A]) \rangle = t_1[AB_x].$$

This shows that  $r$  satisfies  $B_y B_x \subseteq AB_x$ .

Now assume that  $r$  satisfies  $B_y B_x \subseteq AB_z$ . For any tuple  $t \in r$ , there is a tuple  $t_1 \in r$  with  $t[B_y B_x] = t_1[AB_z]$ . Therefore

$$t_1[A] = f_y(t[A]) \quad \text{and} \quad f_x(t[A]) = f_z(t_1[A]).$$

By substituting  $f_y(t[A])$  for  $t_1[A]$  in the right hand equation above, we obtain

$$f_x(t[A]) = f_z(t_1[A]) = f_z(f_y(t[A])).$$

Since this holds for all  $t[A]$ , i.e., all elements of the domain of  $f_x, f_y$ , and  $f_z$ , we can conclude that  $f_x = f_y \circ f_z$ . Thus  $r$  satisfies  $x = y \circ z$ . ■

If  $\tau$  is the composition equation  $x = y \circ z$ , then we call  $B_y B_x \subseteq AB_z$  the *dependency translation* of  $\tau$  and write  $\text{Trans}(\tau) = B_y B_x \subseteq AB_z$ . If  $T$  is a set of composition equations, then  $\text{Trans}(T)$  is the set of dependency translations of equations from  $T$ . Lemma 4 shows that a relational interpretation  $r$  satisfies  $\tau$  iff  $r$  satisfies  $\text{Trans}(\tau)$ . We now have

THEOREM 2. *The implication and finite implication problems for functional dependencies and inclusion dependencies are recursively unsolvable.*

The theorem is a simple consequence of the undecidability results of Post (1947) and Gurevich (1966), as follows. Let  $T \cup \{\tau\}$  be a set of composition equations written using variables from  $U \subseteq \mathcal{X}$ , let  $\Sigma = \text{Trans}(T) \cup \Sigma_U$  and let  $\sigma = \text{Trans}(\tau)$ . It follows from the preceding two lemmas that every functional interpretation satisfying  $T$  also satisfies  $\tau$  iff every relation satisfying  $\Sigma$  also satisfies  $\sigma$ . Similarly, every finite functional interpretation satisfying  $T$  also satisfies  $\tau$  iff every finite relation satisfying  $\Sigma$  also satisfies  $\sigma$ .

## 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 Casanova *et al.* (1982) that no such system exists. Essentially, the difficulties discussed there 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 implications for finite monoids are not recursively enumerable, there is no complete, recursively enumerable axiomatization for finite database 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 (Kanellakis, Cosmadakis, and Vardi, 1983). For example, as suggested in Casanova *et al.* (1982), 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 Kanellakis *et al.* (1983) 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 Kanellakis *et al.* (1983), along with a complete axiom system for finite implication.

The translation presented in Section 5 of monoid equations into depen-

dencies 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 Kanellakis *et al.* (1983) cannot be extended even to *binary* inclusion dependencies.

#### ACKNOWLEDGMENTS

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

RECEIVED: February 4, 1983; ACCEPTED: May 10, 1983

#### REFERENCES

- ARMSTRONG, W. W. (1974), Dependency structures of database relationships, in "Proceeding, IFIP" pp. 580–583, North-Holland, Amsterdam.
- BEERI, C., AND BERNSTEIN, P. A. (1979), Computational problems related to the design of normal form relational schemes, *ACM Trans. Database Systems* **4**(1), 30–59.
- BEERI, C., AND KORTH, H. F. Compatible attributes in a universal relation, in "Proceeding, 1st ACM Conf. on Principles of Database Systems."
- BEERI, C., FAGIN, R., AND HOWARD, J. H. (1977), A complete axiomatization for functional and multivalued dependencies in database relations, in "Proceeding, ACM SIGMOD Conference," pp. 47–61.
- CASANOVA, M. A., FAGIN, R., AND PAPADIMITRIOU, C. H. (1982), Inclusion dependencies and their interaction with functional dependencies, in "Proceeding 1st ACM Conf. on Principles of Database Systems, pp. 171–176; IBM Research Report, RJ3380 (40416); *J. Comput. System Sci.*, in press.
- CHANDRA, A. AND VARDI, M. (1983), "The Implication Problem for Functional and Inclusion Dependencies is Undecidable," Technical Report, RC9980, IBM.
- CHEN, P. (1976), The entity-relationship model—Toward a unified view of data, *ACM Trans. Database Systems* **1** (1), 9–36.
- CODD, E. F. (1980), Extending the database relational model to capture more meaning, *ACM Trans. Database Systems* **4** (4), 397–434.
- FAGIN, R. (1981), A normal form for relational databases that is based on domains and keys, *ACM Trans. Database Systems* **6** (3), 387–415.
- GAREY, M. R., AND JOHNSON, D. S. (1979), "Computers and Intractability: A Guide to the Theory of NP-Completeness," Freeman, San Francisco.
- GUREVICH, Y., AND LEWIS, H. R. (1982), "The Word Problem for Cancellation Semigroups with Zero," Technical Report, TR-08-82, Harvard Univ., Cambridge, Mass.
- GUREVICH, Y. (1966), The word problem for certain classes of semigroups, *Algebra and Logic* **5** (5), 25–35. [Russian]
- JOHNSON, D. S., AND KLUG, A. (1982), Testing containment of conjunctive queries under functional and inclusion dependencies, in "Proceeding, 1st ACM Conf. on Principles of Database Systems," pp. 164–169.
- KANELLAKIS, P. C., COSMADAKIS, S. S., AND VARDI, M. Y. (1983), Unary inclusion dependencies have polynomial time inference problems, in "Proceeding, 15-th ACM Symposium on Theory of Computing," pp. 264–277.
- MACHTEY, M., AND YOUNG, P. (1978), "An Introduction to the General Theory of Algorithms," North-Holland, Amsterdam.

- MITCHELL, J. C. (1983), Inference rules for functional and inclusion dependencies, in "Proceeding, 2nd ACM Conf. on Principles of Database Systems," pp. 58-69.
- POST, E. L. (1947), Recursive unsolvability of a problem of Thue, *J. Symbolic Logic* **12**, 1-11.
- VARDI, M. Y. (1982), "The Implication and Finite Implication Problems for Typed Template Dependencies," Technical Report, STAN-CS-82-912, Stanford Univ. Calif.
- YANNAKAKIS, M., AND PAPADIMITRIOU, C. H. (1982), Algebraic dependencies, *J. Comput. System Sci.* **25** (1), 2-41.