SIAM J. COMPUT.
Vol. 13, No. 1, February 1984


0097-5397/84/1301-0006 $01.25/0

# FORMAL SYSTEMS FOR TUPLE AND EQUALITY GENERATING DEPENDENCIES*

C. BEERI† AND M. Y. VARDI†‡

**Abstract.** We develop several formal systems for tuple and equality generating dependencies. There are three kinds of systems, based upon substitution, tuple elimination and transitivity. We specialize our systems to several subclasses: total dependencies, template dependencies and binary dependencies. We also show that finding a formal system for embedded multivalued dependencies is equivalent to solving the implication problem for that class.

**Key words.** database, dependency, implication, formal system

**1. Introduction.** One of the important issues in the design of relational database schemes is the specification of the constraints that the data must satisfy to model correctly the part of the world under consideration.

Of particular interest are the constraints called *data dependencies*. The first dependencies to be studied were the *functional dependencies* [Codd], which were followed by the *multivalued dependencies* [Del], [Fag1], [Zan]. Later, several types of dependencies were investigated in the literature. Recently, several researchers have independently proposed a new type of dependencies, *tuple* and *equality generating dependencies*, which generalizes all previously studied types [BV2], [Fag2], [YP].[1] Intuitively, the meaning of such a dependency is that if some tuples, satisfying certain equalities, exist in the database, then some other tuples must also exist in the database, or some values in the given tuples must be equal. We assume that the database is many-sorted, i.e., different attributes have different underlying domains.

The *implication problem* for dependencies is to decide whether a given dependency is logically implied by a given set of dependencies. This problem is recursively unsolvable in general [BV2], [CLM], and is solvable but computationally intractable if all tuple generating dependencies are total [BV2], [BV4], [CLM]. A proof procedure for implication of dependencies, called "chase", was studied in [BV3] generalizing [ABU], [MMS] (similar procedures were studied in [Pa], [SU1], [YP]).

The chase enables us to semi-decide implication. In contrast, a *formal system* for dependencies enables us to *derive* new dependencies from the given ones. The interest in formal systems has many aspects. First, a formal system provides us with tools to operate on dependencies, e.g., for equivalence preserving transformations [Ma]. Secondly, a formal system may lead us to discover efficient decision procedures to the implication problem. For example, the formal system for functional dependencies of [Arm] has led to a linear time decision procedure for this class [BB], and the formal system for deriving multivalued dependencies from functional and join dependencies has led to a quadratic time decision procedure for this case [Va2]. Finally, a formal system helps us to gain insight into the dependencies. This insight can lead to useful application, e.g., synthesis of database schemes from functional dependencies [Be]. Formal systems for dependencies have attracted a lot of attention in the last few years, see for example [Arm], [BFH], [BV1], [PJ], [Sc], [SU1], [SU2], [Va1], [YP].

* Received by the editors May 7, 1981, and in revised form November 5, 1982.

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

‡ The research of this author was partially supported by grant 1849/79 of the U.S.A.–Israel Binational Science Foundation. Current address: IBM Research Laboratory, San Jose, California 95193.

[1] We use here the terminology of [BV2], which is different from those of [Fag2], [YP].

It has been demonstrated that dependencies are actually equivalent to sentences in first-order logic [Nic], and in fact the chase is a variant of a well-known theorem-proving procedure: refutation by resolution and paramodulation [BV3]. Thus, it might be argued, there is no need to develop formal systems for dependencies, since any formal system for first-order logic will do. However, dependencies are just a fragment of first-order logic, a fragment that seems to be suitable to expressing integrity constraints of databases, and we would like to have a formal system which would enable us to infer only dependencies and not general first-order sentences, unlike a formal system for first-order logic. Finding formal system for fragments of first-order logic is an active research area in mathematical logic, e.g., a formal system for equations [Bir] and a formal system for equational implications [Sel].

The basic operations in our formalism are replacing a relation by its image under some mapping and replacing the image by the source relation. The first operation is called *instantiation* in traditional theorem-proving terminology [CL]. While these operations allow succinct description of complex derivations, they can be replaced by the much simpler operations of duplicating a tuple (the equivalent of an atomic formula) and renaming variables. We do not pursue this point of view in this paper, but the interested reader can easily translate all our formal systems into that equivalent formalism.

In a formal proof we have *facts* and *rules*. The facts represents specific knowledge relevant to a particular case. The rules express general knowledge about a particular subject area and are used as production rules to generate new fact from old ones. The task of the system is to prove a goal fact from the given ones. Basically, there are two kinds of systems. In *forward* systems, the rules operates on the given fact until a termination condition involving the goal fact is achieved. This is also called a bottom-up search. In *backward* systems, the rules operate on the goal fact until a termination condition involving the given facts is achieved. This is also called top-down search. As an example consider refutation of Horn sets [HW]. Positive unit refutation is an example of a forward system, and input refutation is an example of a backward system. In this paper we have three systems. One of them is a forward system, and the other two are backward systems. We consider one of the backward systems to be highly unnatural for reasons to be discussed later.

Though our main interest is in the full class of tuple and equality generating dependencies, several subclasses are also of interest, e.g., total tuple generating dependencies, template dependencies and embedded multivalued dependencies. We address the problem of specializing our formal systems to such subclasses. That is, when the given dependencies and the goal dependency are all of some subclass, we want all dependencies in the derivation to be of that subclass. Not all subclasses of interest are known to have such a specialized formal system.

It is known that equality can be eliminated from first-order logic by adding the equality axioms: reflexivity, symmetry, transitivity, and substitutivity. This can also be applied to one-sorted dependencies [BV2]. Since the identity relation is not many-sorted, e.g., it is reflexive, we can not eliminate equality from our dependencies which are many-sorted. Nevertheless, in [BV3] it is shown that the role of equality can be "minimized" in deciding implication. Hence, our approach is to develop first formal system for tuple generating dependencies, and then, using a theorem of [BV3], to extend these systems to equality generating dependencies.

The outline of the paper is as follows. In § 2 we define the relational model, tableaux and dependencies, and we describe the chase procedure to test implication of dependencies. In § 3 we develop formal systems for total tuple and equality

generating dependencies, viewing a total tuple generating dependency as a tableau operator. In § 4 we generalize these systems to tuple generating dependencies. In § 5 we address the problem of specializing our systems to several subclasses, and we show that finding a formal system for embedded multivalued dependencies is equivalent to solving the implication problem for this subclass. We also show how to "decompose" a total tuple generating dependency to a set of "weaker" dependencies. We conclude with several remarks in § 6.

## 2. Basic definitions.

**2.1. Attributes, tuples and relations.** *Attributes* are symbols taken from a given finite set $U$ called the *universe*. All sets of attributes are subset of the universe. We use the letters $A, B, C, \cdots$ to denote attributes and $X, Y, \cdots$ to denote sets of attributes. We do not distinguish between the attribute $A$ and the set $\{A\}$. The union of $X$ and $Y$ is denoted by $XY$, and the complement of $X$ in the universe is denoted by $\bar{X}$.

With each attribute $A$ is associated an infinite set called its *domain*, denoted $\mathrm{DOM}(A)$. The domains of distinct attributes are assumed to be disjoint. The domain of a set of attributes $X$ is $\mathrm{DOM}(X) = \bigcup_{A \in X} \mathrm{DOM}(A)$. An *X-value* is a mapping $w: X \to \mathrm{DOM}(X)$, such that $w(A) \in \mathrm{DOM}(A)$ for all $A \in X$. An *X-relation* is a non-empty set (not necessarily finite) of $X$-values. If $X = U$ then we may omit it for simplicity. A *tuple* is a $U$-value. We use $a, b, c, \cdots$ to denote elements of the domains, $s, t, u, \cdots$ to denote tuples, and $I, J, \cdots$ to denote relations.

For a tuple $w$ and a set $Y \subseteq U$ we denote the restriction of $w$ to $Y$ by $w[Y]$. We do not distinguish between $w[A]$, which is an $A$-value, and $w(A)$, which is an element of $\mathrm{DOM}(A)$. Let $I$ be an $X$-relation, and let $Y \subseteq X$. Then the *projection* of $I$ on $Y$, denoted $I[Y]$, is a $Y$-relation $I[Y] = \{w[Y]: w \in I\}$. The set of all attribute values in an $X$-relation $I$ is $\mathrm{VAL}(I) = \bigcup_{A \in X} I[A]$. For an $X$-value $w$, $\mathrm{VAL}(w)$ stands for $\mathrm{VAL}(\{w\})$.

**2.2. Tableaux.** A *valuation* is a partial mapping $h: \mathrm{DOM}(U) \to \mathrm{DOM}(U)$ such that if $h(a)$ is defined then $h(a) \in \mathrm{DOM}(A)$ for all $A \in U$ and $a \in \mathrm{DOM}(A)$. The valuation $h$ can be extended to tuples and relations as follows. Let $w$ be a tuple; then $h(w)$ is $h \circ w$ (i.e., $h$ composed with $w$). Let $I$ be a relation; then $h(I) = \{h(w): w \in I\}$. We say that $\alpha$ is a valuation on a tuple $w$ (a relation $I$) if $\alpha$ is defined exactly on $\mathrm{VAL}(w)$ ($\mathrm{VAL}(I)$). Let $\alpha$ be a valuation on a relation $I$, and let $J$ be a relation. An *extension* of $h$ to $J$ is a valuation on $I \cup J$ that agrees with $h$ on $\mathrm{VAL}(I)$.

A *tableau* [ASU] is a pair $T = \langle w, I \rangle$, where $w$ is a tuple and $I$ is a finite relation, such that $\mathrm{VAL}(w) \subseteq \mathrm{VAL}(I)$. $T$ defines an operation on relation as follows:

$$T(J) = \{h(w): h \text{ is a valuation s.t. } h(I) \subseteq J\};$$

i.e., $T(J)$ is the set of images of $w$ under all valuations that map every tuple of $I$ to some tuple of $J$. Observe that $J \subseteq T(J)$.

*Example* 1. Let $U = \{A, B\}$, $\mathrm{DOM}(A) = \{a0, a1, \cdots\}$, and $\mathrm{DOM}(B) = \{b0, b1, \cdots\}$. Let $I$ be the relation $\{w1, w2, w3\}$:

| | $A$ | $B$ |
|---|---|---|
| $w1$: | $a0$ | $b1$ |
| $w2$: | $a1$ | $b1$ |
| $w3$: | $a1$ | $b0$ |

Let $w$ be the tuple

| $A$ | $B$ |
|---|---|
| $a0$ | $b0$ |

.

Now $T = \langle w, I \rangle$ is a tableau. Let $J$ be the relation:

| $A$ | $B$ |
|---|---|
| $a0$ | $b0$ |
| $a1$ | $b0$ |
| $a1$ | $b1$ |
| $a2$ | $b1$ |
| $a2$ | $b3$ |

.

$T(J)$ is the relation:

| $A$ | $B$ |
|---|---|
| $a0$ | $b0$ |
| $a1$ | $b0$ |
| $a1$ | $b1$ |
| $a2$ | $b1$ |
| $a2$ | $b3$ |
| $a0$ | $b1$ |
| $a1$ | $b3$ |
| $a2$ | $b0$ |

.

Clearly, the values in a tableau serve as formal variables, and therefore can be consistently renamed.

LEMMA 2.1 [ASU]. *Let $\langle w, I \rangle$ be a tableau, and let $h$ be a one-to-one valuation on $I$. Then, for every relation $J$, we have $\langle w, I \rangle(J) = \langle h(w), h(I) \rangle(J)$.*

Let $w$ be any tuple, and consider the tableau $\langle w, \{w\} \rangle$. Clearly $\langle w, \{w\} \rangle(I) = I$ for any relation $I$; i.e. $\langle w, \{w\} \rangle$ defines the identity operation. We will denote this tableau as **1**.

We now show that the set of tableau operations is actually a monoid by demonstrating that it is closed under composition.[2] Let $I$ be a relation and let $u$ and $v$ be tuples; then $I(u/v)$ is the result of substituting $v$ for $u$ in $I$. Formally, we define a one-to-one valuation $h$ on $I \cup \{u\}$ as follows: $h(u[A]) = v[A]$ for all $A \in U$, and $h$ is the identity on all other values. Now we define $I(u/v)$ as $h(I)$. If $\langle u, I \rangle$ is a tableau and $\text{VAL}(v) \cap \text{VAL}(I) = \varnothing$, then by Lemma 2.1 for any relation $K$, $\langle u, I \rangle(K) = \langle v, I(u/v) \rangle(K)$. Let $I$ be a finite nonempty relation, $I = \{w_1, \cdots, w_m\}$. $I$ can be viewed as a mapping from tableaux to relations as follows. Let $\langle u, J \rangle$ be a tableau. We first form $m$ distinct copies of $\langle u, J \rangle$: $\langle u_1, J_1 \rangle, \cdots, \langle u_m, J_m \rangle$, by renaming of values so that no value from $\{u_i\} \cup J_i$ occurs either in $I$ or in $\{u_j\} \cup J_j$, if $i \neq j$. Now $I(u, J)$ is $\bigcup_{i=1}^{m} J_i(u_i/w_i)$.

LEMMA 2.2. *For any relation $K$:*

(1) *If $h$ is a valuation such that $h(I(u, J)) \subseteq K$, then $h(I) \subseteq \langle u, J \rangle(K)$.*

(2) *If $h$ is a valuation on $I$ such that $h(I) \subseteq \langle u, J \rangle(K)$, then there is an extension $h'$ of $h$ to $I(u, J)$ such that $h'(I(u, J)) \subseteq K$.*

*Proof.*

(1) Assume that $h(I(u, J)) \subseteq K$. Then $h(J_i(u_i/w_i)) \subseteq K$, for $1 \leqq i \leqq m$. Thus, $h(w_i) \in \langle w_i, J_i(u_i/w_i) \rangle(K) = \langle u, J \rangle(K)$. I.e., $h(I) \subseteq \langle u, J \rangle(K)$.

[2] This result was independently shown in [FMUY] by a different technique.

(2) Assume that $h$ is a valuation on $I$ such that $h(I) \subseteq \langle u, J\rangle(K)$. Thus, there are valuations $h_1, \cdots, h_m$ such that $h_i(w_i) = h(w_i)$ and $h_i(J_i(u_i/w_i)) \subseteq K$. But $\langle u_i, J_i\rangle$ and $\langle u_j, J_j\rangle$ share no value if $i \neq j$, so there is a valuation $h'$ on $I(u, J)$ which agrees with $h_i$ on $J_i(u_i/w_i)$. It follows that $h'$ is an extension of $h$ and $h'(I(u, J)) \subseteq K$. □

LEMMA 2.3. *Let $\langle w, I\rangle$ and $\langle u, J\rangle$ be tableaux. Then, for every relation $K$, $\langle w, I\rangle(\langle u, J\rangle(K)) = \langle w, I(u, J)\rangle(K)$.*

*Proof.* Let $t \in \langle w, I(u, J)\rangle(K)$. That is, there is a valuation $h$ on $I(u, J)$ such that $h(I(u, J)) \subseteq K$ and $h(w) = t$. By Lemma 2.2, $h(I) \subseteq \langle u, J\rangle(K)$, so $t \in \langle w, I\rangle(\langle u, J\rangle(K))$.

Let $t \in \langle w, I\rangle(\langle u, J\rangle(K))$. That is, there is a valuation $h$ on $I$ such that $h(I) \subseteq \langle u, J\rangle(K)$ and $h(w) = t$. By Lemma 2.2 there is an extension $h'$ of $h$ to $I(u, J)$ such that $h'(I(u, J)) \subseteq K$, so $t \in \langle w, J(u, J)\rangle(K)$. □

We denote $\langle w, I(u, J)\rangle$ by $\langle w, I\rangle \circ \langle u, J\rangle$.

*Example* 1 (continued). To construct $T \circ T = T^2$ we first form three distinct copies of $T$:

| | $A$ | $B$ |
|---|---|---|
| $u_1$: | $a2$ | $b2$ |

| | $A$ | $B$ |
|---|---|---|
| $u_2$: | $a4$ | $b4$ |

| | $A$ | $B$ |
|---|---|---|
| $u_3$: | $a6$ | $b6$ |

| | $A$ | $B$ |
|---|---|---|
| | $a2$ | $b3$ |
| $J_1$: | $a3$ | $b3$ |
| | $a3$ | $b2$ |

| | $A$ | $B$ |
|---|---|---|
| | $a4$ | $b5$ |
| $J_2$: | $a5$ | $b5$ |
| | $a5$ | $b4$ |

| | $A$ | $B$ |
|---|---|---|
| | $a6$ | $b7$ |
| $J_3$: | $a7$ | $b7$ |
| | $a7$ | $b6$ |

Now we form $J_i(u_i/w_i)$:

| $J_1(u_1/w_1)$ | |
|---|---|
| $A$ | $B$ |
| $a0$ | $b3$ |
| $a3$ | $b3$ |
| $a3$ | $b1$ |

| $J_2(u_2/w_2)$ | |
|---|---|
| $A$ | $B$ |
| $a1$ | $b5$ |
| $a5$ | $b5$ |
| $a5$ | $b1$ |

| $J_3(u_3/w_3)$ | |
|---|---|
| $A$ | $B$ |
| $a1$ | $b7$ |
| $a7$ | $b7$ |
| $a7$ | $b0$ |

Now we get that $T^2 = \langle w, J\rangle$, where $J$ is:

| $A$ | $B$ |
|---|---|
| $a0$ | $b3$ |
| $a3$ | $b3$ |
| $a3$ | $b1$ |
| $a1$ | $b5$ |
| $a5$ | $b5$ |
| $a5$ | $b1$ |
| $a1$ | $b7$ |
| $a7$ | $b7$ |
| $a7$ | $b0$ |

Let $T_1$, $T_2$ be tableaux. We say that $T_1$ is *covered* by $T_2$, denoted $T_1 \leqq T_2$, if $T_1(I) \subseteq T_2(I)$, for every relation $I$.

LEMMA 2.4. [ASU] *Let $\langle u, I\rangle$ and $\langle v, J\rangle$ be tableaux. The following conditions are equivalent:*

(1) $\langle u, I\rangle \leqq \langle v, J\rangle$.

(2) $u \in \langle v, J\rangle(I)$.

(3) *There is a valuation $h$ on $J$ such that $h(J) \subseteq I$ and $h(v) = u$.*

*Proof.*

(1) $\Rightarrow$ (2). Assume that $\langle u, I\rangle \leqq \langle v, J\rangle$. Then $\langle u, I\rangle(I) \subseteq \langle v, J\rangle(I)$. But clearly $u \in \langle u, I\rangle(I)$, so $u \in \langle v, J\rangle(I)$.

(2) $\Rightarrow$ (3). Assume that $u \in \langle v, J\rangle(I)$. By definition there is a valuation $h$ on $J$ such that $h(J) \subseteq I$ and $h(v) = u$.

(3) $\Rightarrow$ (1). Assume that $h(J) \subseteq I$ and $h(v) = u$. Let $t \in \langle u, I\rangle(K)$. I.e., there is a valuation $g$ on $I$ such that $g(I) \subseteq K$ and $g(u) = t$. But then $g \circ h$ is a valuation on $J$ such that $g \circ h(J) \subseteq K$ and $g \circ h(v) = g(u) = t$, so $t \in \langle v, J\rangle(K)$. □

Put otherwise, $\langle u, I\rangle \leqq \langle v, J\rangle$ iff there are a valuation $h$ on $J$ and a relation $I'$, such that $u = h(v)$ and $I = h(J) \cup I'$. Thus, covering of tableaux can be easily axiomatized, answering a question posed by [ASU]. Searching for the appropriate valuation $h$ can be quite difficult, since testing covering of tableaux is NP-complete [ASU], [BV4].

*Example* 1 (continued). To show that $T \leqq T^2$, we compute $T^2(I) = T(T(I))$ and get:

| $A$ | $B$ |
|---|---|
| $a0$ | $b1$ |
| $a1$ | $b1$ |
| $a1$ | $b0$ |
| $a0$ | $b0$ |

Now $w \in T^2(I)$, so $T \leqq T^2$. However $T(J)$ is:

| $A$ | $B$ |
|---|---|
| $a0$ | $b3$ |
| $a3$ | $b3$ |
| $a3$ | $b1$ |
| $a1$ | $b5$ |
| $a5$ | $b5$ |
| $a5$ | $b1$ |
| $a1$ | $b7$ |
| $a7$ | $b7$ |
| $a7$ | $b0$ |
| $a0$ | $b1$ |
| $a1$ | $b1$ |
| $a1$ | $b0$ |

Now we have that $w \notin T(J)$, so $T \leqq T^2$. We leave it to the reader to check that the powers of $T$ form a strictly increasing (with respect to containment) sequence of tableaux.

**2.3. Dependencies.** For any given application only a subset of all possible relations is of interest. This subset is defined by constraints which are to be satisfied in the relations of interest. A class of constraints that was extensively studied is the class of dependencies.

A *tuple generating dependency* (tgd) says that if some tuples, satisfying certain equalities exist in the relation, then some other tuples (possibly with some unknown values), must also exist in the relation. Formally, a tgd is a pair of finite relations $\langle I', I\rangle$. It is satisfied by a relation $J$ if for every valuation $h$ on $I$ such that $h(I) \subseteq J$ there is an extension $h'$ of $h$ to $I'$ so that $h'(I') \subseteq J$.

*Multivalued dependencies* (mvd) [Fag1], [Zan] are tgd's of a special form. An mvd is a tgd $\langle\{w\}, I\rangle$, where $|I| \leqq 2$ and $\mathrm{VAL}(w) \subseteq \mathrm{VAL}(I)$. It is usually written $X \twoheadrightarrow Y$, where $X = \{A: |I[A]| = 1\}$ and $Y = \{A: w[A] = u[A]\}$ for some tuple $u \in I$.

An mvd is an example of a *total tuple generating dependency* (ttgd). A tgd $\langle I', I\rangle$ is *total* if $\mathrm{VAL}(I') \subseteq \mathrm{VAL}(I)$. In this case we can assume a simpler form for the tgd.

LEMMA 2.5. [BV3] *Let $\langle I', I\rangle$ be a ttgd, $I' = \{w_1, \cdots, w_m\}$, and let $J$ be a relation; then $J$ satisfies $\langle I', I\rangle$ if and only if it satisfies $\langle\{w_i\}, I\rangle$ for all $1 \leqq i \leqq m$.*

Thus, we can assume without loss of generality that every ttgd is of the form $\langle\{w\}, I\rangle$, and, since $J$ satisfies the ttgd $\langle\{w\}, I\rangle$ iff $\langle w, I\rangle(J) = J$, we will not distinguish between $\langle\{w\}, I\rangle$ and $\langle w, I\rangle$, and treat it both as a tableau operation and a ttgd. The exact meaning will be clear from the context.

A class of dependencies that lies between the class of ttgd's and the class of mvd's is the class of *join dependencies*. We will not deal with join dependencies in this paper. Formal systems for join dependencies are studied in [BV1], [Sc], [Va1].

An *equality generating dependency* (egd) says that, if some tuples satisfying certain equalities exist in the relation, then some values in these tuples must be equal. Formally, an egd is a pair $\langle(a_1, a_2), I\rangle$, where $a_1$ and $a_2$ are $A$-values for some attribute $A$, and $I$ is a finite relation such that $a_1, a_2 \in I[A]$. We also call such an egd an $A$-egd. A relation $J$ satisfies $\langle(a_1, a_2), I\rangle$ if for every valuation $h$ such that $h(I) \subseteq J$ we have $h(a_1) = h(a_2)$. Note that if $a_1 = a_2$ then $\langle(a_1, a_2), I\rangle$ is satisfied by every relation.

*Functional dependencies* (fd) [Codd] are egd's of a special form. An fd is an egd $\langle(a_1, a_2), I\rangle$, where $|I| = 2$ and $\{a_1, a_2\} = I[A]$. It is usually written $X \to A$, where $X = \{B : |I[B]| = 1\}$.

*Example* 2. Let $U = \{A, B, C, D\}$, $\mathrm{DOM}(A) = \{a0, a1, \cdots\}$, $\mathrm{DOM}(B) = \{b0, b1, \cdots\}$ etc. Let $I$ and $J$ be the relations

$I$:

| $A$ | $B$ | $C$ | $D$ |
|---|---|---|---|
| $a0$ | $b0$ | $c1$ | $d0$ |
| $a0$ | $b1$ | $c0$ | $d1$ |

$J$:

| $A$ | $B$ | $C$ | $D$ |
|---|---|---|---|
| $a0$ | $b0$ | $b0$ | $d0$ |
| $a1$ | $b0$ | $c0$ | $d1$ |

Let $u$ and $v$ be the tuples:

| | $A$ | $B$ | $C$ | $D$ |
|---|---|---|---|---|
| $u$: | $a0$ | $b0$ | $c0$ | $d0$ |
| $v$: | $a1$ | $b0$ | $c0$ | $d0$ |

Let $d_1$ be the ttgd $\langle u, I\rangle$. $d_1$ is equivalent to the mvd $A \twoheadrightarrow ABD$. Let $d_2$ be the egd $\langle(a0, a1), J\rangle$. $d_2$ is equivalent to the fd $BC \to A$. $\langle\{u, v\}, I\rangle$ is a tgd.

A dependency is *trivial* if it is satisfied by every relation.

LEMMA 2.6. [BV3]

(1) *The egd $\langle(a_1, a_2), I\rangle$ is trivial if and only if $a_1 = a_2$.*

(2) *The ttgd $\langle w, I\rangle$ is trivial if and only if $w \in I$.*

(3) *The tgd $\langle I', I\rangle$ is trivial if and only if there is a valuation $h$ on $I \cup I'$ which is the identity on $I$ such that $h(I') \subseteq I$.*

**2.4. Implication of dependencies.** For a set of dependencies $D$ we denote by $\mathrm{SAT}(D)$ the set of relations that satisfy all dependencies in $D$. *$D$ implies* a dependency $d$, denoted $D \models d$, if $\mathrm{SAT}(D) \subseteq \mathrm{SAT}(d)$. That is, if $d$ is satisfied for every relation which satisfies all dependencies in $D$. The *implication problem* is to decide for a given set of dependencies $D$ and a dependency $d$ whether $D \models d$. In general the implication problem is recursively unsolvable [BV2], [CLM]. If, however, $D$ consists of egd's and ttgd's then the problem is solvable. A proof procedure[3] for the implication problem—

[3] We distinguish between a *decision procedure* which always halts, and a *proof procedure* which may run forever if the answer to the decision problem is negative.

the chase—was developed in [BV3] generalizing [ABU], [MMS]. (Similar procedures were studied by [Pa], [SU1], [YP].) In the case that $D$ consists of egd's and ttgd's this procedure is a decision procedure.

In the sequel we use $D$ to denote finite sets of dependencies, and we use $d$ and $e$ to denote single dependencies.

Intuitively, to test whether $D$ implies $\langle I', I\rangle$ (or $\langle (a_1, a_2), I\rangle$), we "chase" $I$ by $D$ into SAT $(D)$ and then check if $I'$ is in $I$ (or if $a_1$ and $a_2$ are identical in $I$). Consider first the case that all dependencies are ttgd's. A *chase* of $I$ by $D$ is a sequence of relations $I_0, I_1, \cdots$ such that $I = I_0$ and $I_{j+1}$ is obtained from $I_j$ by an application of a *chase rule*. To each ttgd in $D$ there corresponds a $TT$-rule:

*TT-rule* (for a ttgd $\langle w, J\rangle$ in $D$): $I_{j+1}$ is $\langle w, J\rangle(I_j)$.

*Example* 2 (continued). Consider the ttgd $\langle u, I\rangle$. The effect of the $TT$-rule for this ttgd on a relation $J$ is as follows: if $t_1$ and $t_2$ are tuples in $J$ such that $t_1[A] = t_2[A]$, then add to $J$ the tuple $t$ defined by $t[ABD] = t_1[ABD]$ and $t[C] = t_2[C]$.

We assume that for all $j \geqq 0$, $I_{j+1} \neq I_j$. Thus, the chase is a strictly increasing sequence, and we have:

LEMMA 2.7. [BV3] *All chases of $I$ by $D$ are finite and have the same last relation, which is in SAT $(D)$.* □

This unique last relation is denoted $\text{chase}_D(I)$. It can be used to decide implication.

THEOREM 2.1. [BV3] *Let $D$ be a set of ttgd's, and let $\langle w, I\rangle$ be a ttgd. Then $D \models \langle w, I\rangle$ if and only if $w \in \text{chase}_D(I)$.* □

*Example* 1 (continued). Let $D$ be $\{\langle w, J\rangle\}$, and let $d$ be $\langle w, I\rangle$. To show that $D \models d$, we compute $\text{chase}_D(I)$. $I_0$ is just $I$. $I_1$ is $\langle w, J\rangle(I_0)$:

| $A$ | $B$ |
|---|---|
| $a0$ | $b1$ |
| $a1$ | $b1$ |
| $a1$ | $b0$ |
| $a0$ | $b0$ |

The reader can verify that $\langle w, J\rangle(I_1) = I_1$, so $\text{chase}_D(I) = I_1$ and, since $w \in I_1$, $D \models d$. □

Let us now admit also nontotal tgd's. Trying to generalize our $TT$-rule to tgd's we encounter difficulties, because the new tuples, whose existence in the relation is implied by the existence of some other tuples, are only partly known. The solution is to replace each unknown value by a new distinct value. Let $\langle I', I\rangle$ be a tgd and $h$ a valuation on $I$. A *distinct extension* $h'$ of $h$ to $I'$ is an extension $h'$ of $h$ to $I'$, where, for all $a \in \text{VAL}(I') - \text{VAL}(I)$, $h'(a)$ is a new distinct value. (Since there is an infinite supply of values, we can always choose this new value so as to avoid all possible name clashes.) Our generalized chase rule is now:

> *T-rule* (for some $\langle J', J\rangle$ in $D$): let $h$ be a valuation on $J$ such that $h(J) \subseteq I_j$ but for no extensions $g$ of $h$ to $J'$ we have that $g(J') \subseteq I_j$, and let $f$ be a distinct extension of $h$ to $J'$. $I_{j+1}$ is $I_j \cup f(J')$.

Unlike the $TT$-rules, the $T$-rules are nondeterministic, since they depend on the choice of $h$. Since this rule introduces new values, the chase may be infinite.

*Example* 2 (continued). Consider the tgd $\langle \{v\}, I\rangle$. The effect of the $T$-rule for this tgd on a relation $J$ is as follows: for some tuples $t_1$ and $t_2$ in $J$ such that $t_1[A] = t_2[A]$ but there is no tuple $t$ in $J$ with $t[BD] = t_1$ and $t[C] = t_2$, add such a tuple $t$ to $J$, with $t[A]$ being some new value.

THEOREM 2.2 [BV3]. *Let $D$ be a set of tgd's, and let $\langle I', I\rangle$ be a tgd. Then $D \models \langle I', I\rangle$ if and only if there are a chase of $I$ by $D$: $I_0, I_1, \cdots$, an element $I_n$ of this chase, and a valuation $h$ that is the identity on $I$ such that $h(I') \subseteq I_n$.* □

Let us now admit also egd's. It seems that we need another chase rule for egd's, and indeed in [BV3] such a rule is used. However, it is also shown there how the use of this rule can be avoided.

Let $e$ be an $A$-egd $\langle (a_1, a_2), I\rangle$. Let $w_1$ be a tuple such that $w_1[A] = a_1$, and for all $B \in \bar{A}$, we have $w_1[B] \notin I[B]$. Let $w_2$ be a tuple such that $w_2[A] = a_2$ and $w_2[\bar{A}] = w_1[\bar{A}]$. We associate with $e$ two ttgd's. $e_1$ is $\langle w_1, I \cup \{w_2\}\rangle$, and $e_2$ is $\langle w_2, I \cup \{w_1\}\rangle$. Intuitively, $e_1$ states that, given $I$, wherever $a_2$ appears $a_1$ also appears. More precisely, if a relation contains $h(I)$, then for each tuple in it which contains $h(a_2)$ there exists a tuple identical to it except that $h(a_2)$ is replaced by $h(a_1)$. Similarly, $e_2$ states that wherever $a_1$ appears, $a_2$ also appears. Let $D^*$ be the result of replacing each egd $e$ in $D$ by $e_1$ and $e_2$. The idea is that instead of saying that two values are equal, we say that they "look the same from within the relation".

*Example* 2 (continued). Let $e$ be $\langle (a0, a1), J\rangle$. $e$ stands for the fd $BC \to A$. $w_1$, and $w_2$ are the tuples:

| | $A$ | $B$ | $C$ | $D$ |
|---|---|---|---|---|
| $w_1$: | $a0$ | $b1$ | $c1$ | $d2$ |
| $w_2$: | $a1$ | $b1$ | $c1$ | $d2$ |

$e_1$ is $\langle w_1, J \cup \{w_2\}\rangle$, and $e_2$ is $\langle w_2, J \cup \{w_1\}\rangle$.

THEOREM 2.3 [BV3].

(1) *$e \models e_1$ and $e \models e_2$.*

(2) *Let $d$ be a tgd, $D \models d$ if and only if $D^* \models d$.*

(3) *Let $e$ be an $A$-egd $\langle (a_1, a_2), I\rangle$. Then $D \models e$ if and only if there is a chase of $I$ by $D^*$: $I_0, I_1, \cdots$, an element $I_n$ of this chase, an $A$-egd $\langle (a_3, a_4), J\rangle$ in $D$, and a valuation $h$ on $J$ such that $h(J) \subseteq I_n$, $h(a_3) = a_1$, and $h(a_4) = a_2$.*

(4) *Let $e$ be a nontrivial egd. Then $D \models e$ if and only if $D^* \models e_1$, and there is a nontrivial $A$-egd in $D$.*

We will rely upon Theorems 2.1, 2.2, and 2.3 in developing formal systems for dependencies.

**3. Formal systems for ttgd's and egd's.** A *formal system* for a family of dependencies consists of axioms and inference rules. The axioms are schemas of trivial dependencies, e.g., the reflexivity axiom for fd's [Arm] and mvd's [BFH]. The inference rules specify whether a dependency is inferable from some premises, e.g., the transitivity rule for fd's [Arm] and mvd's [BFH]. If the number of premises in the rule is bounded then the rule is said to be *bounded*. Let $\Psi$ be a class of dependencies, and let $F$ be a formal system. A *derivation* of a dependency $d$ from a set of dependencies $D$ by $F$ in $\Psi$ is a sequence of dependencies from $\Psi$: $d_0, d_1, \cdots, d_n$, with $d_n = d$, each of which is either an instance of an axiom of $F$, a member of $D$, or is inferrable from earlier $d_i$'s by one of the inference rules of $F$. We say that $d$ is *derivable* from $D$ by $F$ in $\Psi$, denoted $D \vdash_{F,\Psi} d$, if there is a derivation of $d$ from $D$ by $F$ in $\Psi$. If $F$ and $\Psi$ are understood from the context, then we simply write $D \vdash d$. $F$ is *sound* for $\Psi$ if for every $D \subseteq \Psi$, $d \in \Psi$ we have that $D \vdash_{F,\Psi} d$ implies $D \models d$, and is *complete for* $\Psi$ if for every $D \subseteq \Psi$, $d \in \Psi$ we have that $D \models d$ implies $D \vdash_{F,\Psi} d$. To show that $F$ is sound suffice it to show that for every $d_i$ in a derivation of $d$ from $D$ in $F$, $D \models d_i$. That is, if $d_i$ is an instance of an axiom then it is trivial, i.e., the axioms are sound, and if $d_i$ is inferable from $d_{j_1}, \cdots, d_{j_m}$ then $\{d_{j_1}, \cdots, d_{j_m}\} \models d_i$, i.e., the inference rules are sound.

There are two ways of looking at inference rules. An inference rule can be defined as a recursive predicate saying whether a dependency is inferable from the premises, or it can be defined as a recursive function giving the inferred dependency in answer to the given premises. In the remaining parts of this section we exhibit several formal systems for the family of ttgd's and egd's, usually viewing inference rules as predicates. In § 4 we exhibit formal systems for the family of tgd's and egd's, usually viewing inference rules as functions.

**3.1. Tableau composition.** Our first formal system consists of one axiom and two inference rules. Its completeness is based upon the following lemma.

LEMMA 3.1. *Let $D$ be a set of ttgd's, and let $T$ be a ttgd. Then $D \models T$ if and only if there is a sequence $T_1, \cdots, T_n$, $n \geqq 0$, of tableaux from $D$ such that $T_n \circ \cdots \circ T_1 \geqq T$. (For $n = 0$ the composition is defined as* 1.)

*Proof.* $D \models \langle w, I\rangle$ iff (by Theorem 2.1) $w \in \text{chase}_D(I)$ iff there is a sequence $T_1, \cdots, T_n$, $n \geqq 0$, of tableaux from $D$ such that $w \in \text{chase}_D(I) = T_n(\cdots(T_1(I))\cdots) = T_n \circ \cdots \circ T_1(I)$ iff (by Lemma 2.4) $T_n \circ \cdots \circ T_1 \geqq \langle w, I\rangle$. □

We present now the system $TT_1$:

TTD0 (triviality): $\vdash \langle w, \{w\}\rangle$.

TTD1 (covering): $\langle u, I\rangle \vdash \langle v, J\rangle$ if $\langle v, J\rangle \leqq \langle u, I\rangle$.

TTD2 (composition): $\langle u, I\rangle, \langle v, J\rangle \vdash \langle u, I\rangle \circ \langle v, J\rangle$.

TTD0 is analogous to the $J$-axiom of [BV1], [Va1], TTD1 is analogous to their covering rule, and TTD2 is analogous to their projection-substitution rule.

THEOREM 3.1. *The system $TT_1$ is sound and complete for ttgd's.*

*Proof.*

*Soundness.* $\langle w, \{w\}\rangle$ is a trivial ttgd by Lemma 2.6.

Suppose that $\langle v, J\rangle \leqq \langle u, I\rangle$ and $K \in \text{SAT}(\langle u, I\rangle)$. Then $K \subseteq \langle v, J\rangle(K) \subseteq \langle u, I\rangle(K) = K$. So $K \in \text{SAT}(\langle v, J\rangle)$, and TTD1 is sound.

Suppose now that $K \in \text{SAT}(\langle u, I\rangle, \langle v, J\rangle)$. Then

$$K \subseteq (\langle u, I\rangle \circ \langle v, J\rangle)(K) = \langle u, I\rangle(\langle v, J\rangle(K)) = \langle u, I\rangle(K) = K.$$

So $K \in \text{SAT}(\langle u, I\rangle \circ \langle v, J\rangle)$, and TTD2 is sound.

*Completeness.* Suppose that $D \models \langle w, I\rangle$. By Lemma 3.1, there is a sequence $T_1, \cdots, T_n$, $n \geqq 0$, of tableaux from $D$ such that $T_n \circ \cdots \circ T_1 \geqq \langle w, I\rangle$. By $n-1$ applications of TTD2 (or one application of TTD0) we get $D \vdash T_n \circ \cdots \circ T_1$, and applying TTD1 we get $D \vdash \langle w, I\rangle$. □

**3.2. Tableau simplification.** Rule TTD2 enables us to derive a "big" ttgd from "smaller" ones. In this section we introduce a rule which enables us to derive from given ttgd's a ttgd of reduced size.

We now present the system $TT_2$, which has one axiom and one inference rule:

TTD0′ (triviality): $\vdash \langle w, \{w\} \cup I\rangle$.

TTD3 (simplification): $\langle u, I\rangle, \langle v, \langle u, I\rangle(J)\rangle \vdash \langle v, J\rangle$.

Rule TTD0′ is a stronger version of TTD0. By Lemma 2.6, it characterizes all trivial ttgd's. Rule TTD3 has no analogue in the formal systems for join dependencies in [BV1], [Sc], [Va1].

THEOREM 3.2. *The system $TT_2$ is sound and complete for ttgd's.*

*Proof.*

*Soundness.* $\langle w, \{w\} \cup I\rangle$ is a trivial ttgd by Lemma 2.6. To prove that rule TTD3 is sound we have to show that $D \models \langle v, J\rangle$, if $D = \{\langle u, I\rangle, \langle v, \langle u, I\rangle(J)\rangle\}$. We compute a chase of $J$ by $D$. $J_0$ is $J$, $J_1$ is $\langle u, I\rangle(J_0) = \langle u, I\rangle(J)$, and $J_2$ is $\langle v, \langle u, I\rangle(J)\rangle(J_1) = \langle v, \langle u, I\rangle(J)\rangle(\langle u, I\rangle(J))$. Thus, $v \in J_2 \subseteq \text{chase}_D(I)$, and $D \models \langle v, J\rangle$.

*Completeness.* Suppose that $D \models \langle w, I\rangle$. By Theorem 2.1, there is a sequence $T_1, \cdots, T_n$, $n \geq 0$, of tableaux from $D$ such that $w \in T_n(\cdots(T_1(I)\cdots)$. By TTD0', $D \vdash \langle w, T_n(\cdots(T_1(I)\cdots)\rangle$, and $n$ applications of TTD3 give $D \vdash \langle w, I\rangle$. □

**3.3. Tableau transitivity.** Our third system $TT_3$ consists of rule TTD0' from the previous section and rule TTD4, which is reminiscent of the transitivity rule for fd's and mvd's but unlike them is not a bounded rule. Rule TTD4 is analogous to rule JD5 for join dependencies of [BV1], [Va1].

TTD4 (transitivity). $\langle w, I\rangle, \langle u_1, J\rangle, \cdots, \langle u_m, J\rangle \vdash \langle u, J\rangle$, if there is a valuation $h$ such that

$$h(I) \subseteq \{u_1, \cdots, u_m\} \text{ and } h(w) = u.$$

The condition in the rule can be reformulated as $u \in \langle w, I\rangle(\{u_1, \cdots, u_m\})$.

To prove completeness we use the following observation.

LEMMA 3.2. *Let $D$ be a set of ttgd's, and let $I$ be a finite relation. Then* $\text{chase}_D(I) = \{w : D \models \langle w, I\rangle\}$.

*Proof.* By Theorem 2.1, $w \in \text{chase}_D(I)$ iff $D \models \langle w, I\rangle$. □

THEOREM 3.3. *The system $TT_3$ is sound and complete for ttgd's.*

*Proof.*

*Soundness.* To prove that TTD4 is sound we have to show that $D \models \langle u, J\rangle$, if $D = \{\langle w, I\rangle, \langle u_1, J\rangle, \cdots, \langle u_m, J\rangle\}$, and there is a valuation $h$ such that $h(I) \subseteq \{u_1, \cdots, u_m\}$ and $h(w) = u$. We compute a chase of $J$ by $D$. $J_0$ is $J$, $J_i$ is $\langle u_i, J\rangle(J_{i-1})$, for $1 \leq i \leq m$, and $J_{m+1}$ is $\langle w, I\rangle(J_m)$. Clearly, $\{u_1, \cdots, u_m\} \subseteq J_m$, so $u \in J_{m+1} \subseteq \text{chase}_D(J)$, and $D \models \langle u, J\rangle$.

*Completeness.* By Lemma 3.2, suffice it to show that for every $u \in \text{chase}_D(J)$, $D \vdash \langle u, J\rangle$. Let $J_0, \cdots, J_n$ be a chase of $J$ by $D$. We show by induction on $i$ that for every $u \in J_i$, we have $D \vdash \langle u, J\rangle$. $J_0$ is $J$, so if $u \in J$, then $D \vdash \langle u, J\rangle$ by rule TTD0'. Suppose now that the assumption holds for $J_i = \{u_1, \cdots, u_m\}$. Let $u \in J_{i+1}$. That is, there is a ttgd $\langle w, I\rangle \in D$, and a valuation $h$ such that $h(I) \subseteq J_i$ and $h(w) = u$. By the induction hypothesis, $D \vdash \langle u_k, J\rangle$, for $1 \leq k \leq m$, so $D \vdash \langle u, J\rangle$ by rule TTD4. □

**3.4. ttgd's and egd's.** Using Theorem 2.3 we can easily extend the formal system for ttgd's of the preceding sections to deal with egd's as well.

We first present rules that deal only with egd's.

ED0 (triviality). $\vdash \langle (a, a), I\rangle$, if $a \in \text{VAL}(I)$.

With an eye to Lemma 2.4 we define covering for egd's. An egd $\langle (a_1, a_2), I\rangle$ *covers* an egd $\langle (a_3, a_4), J\rangle$, denoted $\langle (a_3, a_4), J\rangle \leq \langle (a_1, a_2), I\rangle$, if there is a valuation $h$ such that $h(I) \subseteq J$, $h(a_1) = a_3$, and $h(a_2) = a_4$.

ED1 (covering): $\langle (a_1, a_2), I\rangle \vdash \langle (a_3, a_4), J\rangle$, if $\langle (a_3, a_4), J\rangle \leq \langle (a_1, a_2), I\rangle$.

LEMMA 3.3. *Rules* ED0 *and* ED1 *are sound.*

*Proof.* The soundness of ED0 follows from Lemma 2.6, and that of ED1 from Theorem 2.3. □

The next rule enables us to infer ttgd's from egd's. Recall that with each egd $e$ we associate two ttgd's $e_1$ and $e_2$.

ETTD0 (translation): $e \vdash e_1$, $e \vdash e_2$.

LEMMA 3.4. *Let $F$ be any sound and complete formal system for ttgd's, and let $F'$ be $F \cup \{\text{ETTD0}\}$. If $D$ is a set of ttgd's and egd's, then $D \vdash_{F'} \langle w, I\rangle$ if and only if $D \models \langle w, I\rangle$.*

*Proof.*

*Only if.* We have to show that rule ETTD0 is sound, but this follows immediately from Theorem 2.3.

*If.* Suppose that $D \models \langle w, I\rangle$. By Theorem 2.3, $D^* \models \langle w, I\rangle$, so by assumption, $D^* \vdash \langle w, I\rangle$. But $D \vdash D^*$ by rule ETTD0. The claim follows. □

All the formal systems in the sequel include ED0 and ETTD0. Thus, in view of Lemmas 2.6 and 3.4, in order to prove completeness it suffices to consider implication of non-trivial egd's, and we can also use the fact that $D \vdash_{\text{ETTD0}} D^*$ without mentioning it explicitly.

We present now three inference rules analogous to TTD2, TTD3, and TTD4 respectively.

ETTD1 (composition). $\langle (a_1, a_2), I\rangle, \langle u, J\rangle \vdash \langle (a_1, a_2), I(u, J)\rangle$.

ETTD2 (simplification). $\langle u, I\rangle, \langle (a_1, a_2), \langle u, I\rangle (J)\rangle \vdash \langle (a_1, a_2), J\rangle$.

ETTD3 (transitivity). $\langle (a_3, a_4), I\rangle, \langle u_1, J\rangle, \cdots, \langle u_m, J\rangle \vdash \langle (a_1, a_2), J\rangle$, if there is a valuation $h$ such that $h(I) \subseteq \{u_1, \cdots, u_m\}$, $h(a_3) = a_1$, and $h(a_4) = a_2$.

Let $ETT_1$ be the system $TT_1 \cup \{$ED0, ED1, ETTD0, ETTD1$\}$, let $ETT_2$ be the system $TT_2 \cup \{$ED0, ED1, ETTD0, ETTD2$\}$, and let $ETT_3$ be the system $TT_3 \cup \{$ED0, ETTD0, ETTD3$\}$.

THEOREM 3.4. *The system $ETT_1$ is sound and complete for ttgd's and egd's.*

*Proof.*

*Soundness.* We have to show that rule ETTD1 is sound. Let $K \in \text{SAT}(\langle (a_1, a_2), I\rangle, \langle v, J\rangle)$, and suppose that $h(I(v, J)) \subseteq K$. By Lemma 2.2, $h(I) \subseteq \langle v, J\rangle(K) = K$. It follows that $h(a_1) = h(a_2)$ and $K \in \text{SAT}\langle (a_1, a_2), I(v, J)\rangle$.

*Completeness.* Suppose that $D \models \langle (a_1, a_2), I\rangle$. By Theorem 2.3, there is a sequence $T_1, \cdots, T_n$, $n \geqq 0$, of tableaux from $D^*$, an egd $\langle (a_3, a_4), J\rangle$ from $D$, and a valuation $h$ on $J$ such that $h(J) \subseteq T_n(\cdots(T_1(I))\cdots)$, $h(a_3) = a_1$, and $h(a_4) = a_2$. Let $T_n \circ \cdots \circ T_1$ be $\langle u, K\rangle$. By rule TTD2 or TTD0, $D \vdash \langle u, K\rangle$, so by ETTD2, $D \vdash \langle (a_3, a_4), J(u, K)\rangle$. But, by Lemma 2.2, there is an extension $h'$ of $h$ to $J(u, K)$ such that $h'(J(u, K)) \subseteq I$. It follows that

$$\langle (a_1, a_2), I\rangle \leqq \langle (a_3, a_4), J(u, K)\rangle,$$

and by rule ED1, $D \vdash \langle (a_1, a_2), I\rangle$. □

The proofs of Theorems 3.5 and 3.6 are analogous to the proof of Theorem 3.4, and are left to the reader.

THEOREM 3.5. *The system $ETT_2$ is sound and complete for ttgd's and egd's.*

THEOREM 3.6. *The system $ETT_3$ is sound and complete for ttgd's and egd's.*

Up to now we have referred only to clauses (1), (2) and (3) in Theorem 2.3. By referring to clause (4) in that theorem we get rule ETTD4, which is more flexible than ETTD1, ETTD2, and ETTD3 in the sense that it can be combined with any sound and complete system for ttgd's to give a sound and complete system for ttgd's and egd's.

ETTD4: $d, e_1 \vdash e$, if $d$ and $e$ are $A$-egd's, and $d$ is nontrivial.

THEOREM 3.7. *Let $F$ be a sound and complete system for ttgd's, and let $F'$ be $F \cup \{$ED0, ETTD0, ETTD4$\}$. Then $F'$ is a sound and complete system for ttgd's and egd's.*

*Proof.*

*Soundness.* We have to show that rule ETTD5 is sound, but this is immediate from Theorem 2.3.

*Completeness.* Suppose that $D \models e$, where $e$ is a nontrivial $A$-egd. By Theorem 2.3, $D^* \models e_1$, and there is a nontrivial $A$-egd $d$ in $D$. By assumption $D^* \vdash e_1$, so by rules ETTD0 and ETTD4, $D \vdash e$. □

**3.5. Discussion.** We refer now to the forward/backward classification of our formal systems. In deriving $\langle w, I\rangle$ from $D$, $w$ is the goal fact, and the tuples of $I$ are the given facts. The dependencies in $D$ are the production rules. The system $TT_3$ is clearly a forward system. Starting with the tuples of $I$, one generates additional tuples (facts) until the goal tuple is generated. Actually, this is a direct simulation of the chase. In contrast, the system $TT_1$ is a backward system where the rules operate on a collection of goal tuples $K$. Initially, $K$ is just $\{w\}$. The operation of a ttgd $\langle u, J\rangle$ from $D$ on the set of goal tuples is to replace a tuple $v \in K$ by $J(u/v)$. The process terminates when there is a valuation $h$ which is the identity on $w$ such that $h(K) \subseteq I$.

The system $TT_2$ is another type of backward system. It starts with guessing an initial collection of facts that includes also the goal fact. This collection is just $\text{chase}_D(I)$. The operation of a ttgd $\langle u, J\rangle$ from $D$ on this collection of guessed facts is to eliminate facts that are implied by other facts, i.e., replacing $\langle u, J\rangle(K)$ by $K$. The termination condition is that all the remaining facts are given ones, viz., members of $I$. It is the initial guess of all facts that makes this system highly unnatural. While in the other systems one starts from the problem ($w$ or $I$) and then has to guide the production, there seems no natural way of guessing $\text{chase}_D(I)$ "straight from the blue".

These observations extend to the systems $ETT_1$, $ETT_2$, and $ETT_3$.

Let us refer now to the *length* and *size* of the derivations in our formal systems. (By the size of a derivation we mean the number of symbols in the derivation.) If $D \models \langle w, I\rangle$, then for all chases of $I$ by $D$: $I_0, I_1, \cdots, I_n = \text{chase}_D(I)$ we have $w \in I_n$. In [BV3] we show that both $n$ and the size of the $I_j$'s can be exponential in the size of $I$. Thus, it is clear that the derivations that were constructed in the various completeness proofs can be of length and size exponential in the size of $D$ and $I$, since they all simulate the chase, directly or indirectly. Can we construct smaller or shorter derivations?

The answer is probably negative. In [Va3] it is shown that, given a derivation in any of the above systems, one can translate it into a derivation in any of the other systems with at most a polynomial increase in the length and size of the derivation. Furthermore, given a derivation in the system $TT_3$ whose length is polynomial in the size of $D$ and $I$, one can construct another derivation whose size is polynomial in the size of $D$ and $I$. Now, for a sequence $d_0, \cdots, d_n$, one can check in space that is polynomial in the size of the input, whether it is a derivation of $d$ from $D$ by any of the aforementioned formal systems. If we could bound the length of the derivations by a polynomial in the size of $D$ and $I$, it would follow that the set $\{\langle D, d\rangle : D \models d\}$ is in PSPACE. This is quite unlikely, since it is shown in [CLM] that this set is logspace complete in EXPTIME.

**4. Formal systems for tgd's and egd's.** The completeness proof for the systems in the previous section used the fact that $\text{chase}_D(I) = T_n(\cdots(T_1(I)\cdots)$. Allowing tgd's in $D$, that is no longer true. Nevertheless, we will be able to generalize our systems to deal with tgd's, using the essential ideas underlying them.

Let $\langle I', I\rangle$ be a tgd. It can be viewed as an implicational constraint where $I$ serves as the antecedent and $I'$ serves as the consequent, saying roughly, that if $I$ can be "embedded" in a relation $J$ then so can be $I'$.[4] We partition the set of values in $I \cup I'$ into two sets. The set of *existential* values in $\langle I', I\rangle$ is $\text{EX}(I', I) = \text{VAL}(I') - \text{VAL}(I)$, and the set of *universal* values is the set $\text{VAL}(I)$. (The source for this terminology is the way dependencies are written as first-order sentences. See [Fag2], [Va3].) It is

[4] Indeed, tgd's and egd's are called *embedded implicational dependencies* in [Fag2].

clear that existential and universal values play completely different roles in the "meaning" of $\langle I', I\rangle$. If $h$ is a valuation on $I$ then to extend it to $I'$ we have to define it on $\mathrm{EX}(I', I)$.

**4.1. Substitution.** Studying rule TTD2, we observe that the basic operation there is that of replacing, when given $\langle w, I\rangle$ and $\langle u, J\rangle$,a tuple $v$ in $I$ by $J(u/v)$, because the existence of $J(u/v)$ entails the existence of $v$. We generalize that to tgd's.

We present now the system $T_1$:

TD0 (triviality). $\vdash\langle I', I\rangle$, if there is a valuation $h$ which is the identity on $I$ such that $h(I')\subseteq I$.

TD1 (collapsing). $\langle I', I\rangle\vdash\langle h(I'), h(I)\rangle$, if $h$ is a valuation such that $h|_{\mathrm{EX}(I',I)}$ is a one-to-one mapping into $\mathrm{EX}(h(I'), h(I))$.

TD2 (augmentation). $\langle I', I\rangle\vdash\langle I', I\cup J\rangle$, if $\mathrm{EX}(I', I)\cap\mathrm{VAL}(J)=\varnothing$.

TD3 (projection). $\langle I'\cup J, I\rangle\vdash\langle I', I\rangle$.

TD4 (substitution). $\langle I', I\cup J'\rangle$, $\langle J', J\rangle\vdash\langle I', I\cup J\rangle$, if $\mathrm{VAL}(I)\cap\mathrm{VAL}(J')\subseteq\mathrm{VAL}(J)$ and $\mathrm{EX}(I', I\cup J')\cap\mathrm{VAL}(J)=\varnothing$.

Rule TD3 has no analogous rule for ttgd's. Rule TD0 generalizes rule TTD0, rules TD1 and TD2 generalize rule TTD1, and rule TD4 generalizes rule TTD2.

THEOREM 4.1. *The system $T_1$ is sound and complete for tgd's.*

*Proof.*

*Soundness.* TD0 is sound by Lemma 2.6.

Let $J\in\mathrm{SAT}(\langle I', I\rangle)$, let $h$ be a valuation such that $h|_{\mathrm{EX}(I',I)}$ is a one-to-one mapping into $\mathrm{EX}(h(I'), h(I))$, and let $g$ be a valuation on $h(I)$ such that $g(h(I))\subseteq J$. Since $J$ satisfies $\langle I', I\rangle$, there is an extension $f$ of $g\circ h$ to $I'$ such that $f(I')\subseteq J$. We define an extension $g'$ of $g$ to $h(I')$ as follows. Let $a\in\mathrm{EX}(h(I'), h(I))$. Then there is a unique value $a'\in\mathrm{EX}(I', I)$ such that $a=h(a')$. We let $g'(a)=f(a')$. Now $g'(h(I'))=f(I')\subseteq J$, so TD1 is sound.

Let $K\in\mathrm{SAT}(\langle I', I\rangle)$, and suppose that $\mathrm{EX}(I', I)\cap\mathrm{VAL}(J)=\varnothing$. If $h$ is a valuation on $I\cup J$ such that $h(I\cup J)\subseteq K$, then there is an extension $h'$ of $h|_I$ to $I'$ such that $h'(I')\subseteq K$. Clearly, $h'$ is also an extension of $h$ to $I'$, so TD2 is sound.

Let $K\in\mathrm{SAT}(\langle I'\cup J, I\rangle)$, and let $h$ be a valuation on $I$ such that $h(I)\subseteq K$. There is an extension $g$ of $h$ to $I'\cup J$ such that $h(I'\cup J)\subseteq K$. Clearly, $g$ is also an extension of $h$ to $I'$ such that $g(I')\subseteq K$, so TD3 is sound.

Let $K\in\mathrm{SAT}(\langle I', I\cup J'\rangle, \langle J', J\rangle)$, where $\mathrm{VAL}(I)\cap\mathrm{VAL}(J')\subseteq\mathrm{VAL}(J)$ and $\mathrm{EX}(I', I\cup J')\cap\mathrm{VAL}(J)=\varnothing$. Let $h$ be a valuation on $I\cup J$ such that $h(I\cup J)\subseteq K$. Now $\mathrm{EX}(J', J)\cap\mathrm{VAL}(I)=\varnothing$, and by TD2, $K\in\mathrm{SAT}(J', J\cup I)$, so there is an extension $g$ of $h$ to $J'$ such that $g(J')\subseteq K$, and therefore $g(I\cup J')\subseteq K$. $g$ is undefined on $\mathrm{EX}(I', I\cup J')$, because $\mathrm{EX}(I', I\cup J')\cap\mathrm{VAL}(I\cup J\cup J')=\varnothing$. Since $K$ satisfies $\langle I', I\cup J\rangle$, there is an extension $f$ of $g$ to $I'$ such that $f(I')\subseteq K$. $f$ is also an extension of $h$, so TD4 is sound.

*Completeness.* Suppose that $D\models\langle I', I\rangle$. Then there are a chase of $I$ by $D$: $I_0, I_1, \cdots$, an element $I_n$ of this chase, and a valuation $h$ which is the identity on $I$ such that $h(I')\subseteq I_n$. We can assume without loss of generality that $\mathrm{EX}(I', I)\cap\mathrm{VAL}(I_n)=\varnothing$. We construct, by backward induction on $k$ from $n$ to 0, a relation $J_k$ such that $D\vdash\langle I', J_k\rangle$, $J_k\subseteq I_k$, and $\mathrm{EX}(I', J_k)\cap\mathrm{VAL}(I_k)=\varnothing$.

*Basis* $(k=n)$. Let $J_n=h(I')$. By TD0, $D\vdash\langle I', h(I')\rangle$, $h(I')\subseteq I_n$, and $\mathrm{EX}(I', h(I'))\cap\mathrm{VAL}(I_n)=\varnothing$.

*Induction.* Suppose that $D\vdash\langle I', J_{k+1}\rangle$, $J_{k+1}\subseteq I_{k+1}$, and $\mathrm{EX}(I', J_{k+1})\cap\mathrm{VAL}(I_{k+1})=\varnothing$. There are some $\langle K', K\rangle$ in $D$ and a valuation $g$ on $K\cup K'$ such that $g|_{\mathrm{EX}(K',K)}$ is a one-to-one mapping into $\mathrm{EX}(g(K), g(K'))$, $I_{k+1}=I_k\cup g(K')$, $g(K)\subseteq I_k$,

and $\mathrm{EX}(g(K'), g(K)) \cap \mathrm{VAL}(I_k) = \emptyset$. $J_{k+1}$ can be viewed as $J' \cup J''$, where $J' \subseteq I_k$ and $J'' \subseteq g(K')$. By TD1 and TD3, $\langle K', K\rangle \vdash \langle J'', g(K)\rangle$. Now

$$\mathrm{VAL}(J') \cap \mathrm{VAL}(J'') \subseteq \mathrm{VAL}(I_k) \cap \mathrm{VAL}(g(K')) \subseteq \mathrm{VAL}(g(K)),$$

and

$$\mathrm{EX}(I', J_{k+1}) \cap \mathrm{VAL}(g(K)) \subseteq \mathrm{EX}(I', J_{k+1}) \cap \mathrm{VAL}(I_{k+1}) = \emptyset,$$

so by TD4, $D \vdash \langle I', J_k\rangle$, where $J_k = J' \cup g(K)$. Clearly, $J_k \subseteq I_k$. Also

$$\mathrm{EX}(I', J_k) \subseteq \mathrm{EX}(I', J_{k+1}) \cup \mathrm{EX}(g(K'), g(K)),$$

and since $\mathrm{EX}(g(K'), g(K)) \cap \mathrm{VAL}(I_k) = \emptyset$, we have $\mathrm{EX}(I', J_k) \cap \mathrm{VAL}(I_k) = \emptyset$.

In particular, we have $D \vdash \langle I', J_0\rangle$, $J_0 \subseteq I$, and $\mathrm{EX}(I', J_0) \cap \mathrm{VAL}(I) = \emptyset$, so by TD2, $D \vdash \langle I', I\rangle$. □

**4.2. Tuple elimination.** Studying rule TTD3, we observe that the basic operation is that of eliminating, when given $\langle w, I\rangle$ and $\langle u, J\rangle$, a tuple $v$ from $I$, if the tuples of $J(u/v)$ are in $I$, because the existence of the tuples of $J(u/v)$ implies the existence of $v$. We generalize this to tgd's.

We now present the system $T_2$:

TD0 (triviality)

TD1 (collapsing)

TD2 (augmentation)

TD5 (tuple elimination). $\langle I', I \cup J\rangle$, $\langle J, I\rangle \vdash \langle I', I\rangle$.

Rule TD5 is implied by rule TD4. For the simplicity of the system we pay with less natural derivations.

THEOREM 4.2. *The system $T_2$ is sound and complete for tgd's.*

*Proof.*

*Soundness.* The rules are implied by the rules of the system $T_1$, so they are sound.

*Completeness.* Suppose that $D \models \langle I', I\rangle$. Then there are a chase of $I$ by $D$: $I_0, I_1, \cdots$, an element $I_n$ of this chase, and a valuation $h$ which is the identity on $I$ such that $h(I') \subseteq I_n$. We can assume without loss of generality that $\mathrm{EX}(I', I) \cap \mathrm{VAL}(I_n) = \emptyset$. We leave it to the reader to show, by backward induction on $k$ from $n$ to 0, that $D \vdash \langle I', I_k\rangle$. In particular, it follows that $D \vdash \langle I', I\rangle$. □

**4.3. Transitivity.** Studying rule TTD4 we see that it is basically a transitivity rule. The essential idea is that if the existence of a set of tuples $J$ implies the existence of a set of tuples $J'$, and the existence of $J'$ implies the existence of a tuple $w$, then the existence of $J$ implies the existence of $w$. We generalize this to tgd's. It turns out that in contrast to TTD4 the generalized transitivity rule is a bounded rule.

We present now the system $T_3$:

TD0′ (triviality). $\vdash \langle I, I\rangle$.

TD1 (collapsing)

TD2 (augmentation)

TD3 (projection)

TD6 (weakening). $\langle h(I'), I\rangle \vdash \langle I', I\rangle$, if $h$ is the identity on $I$.

TD7 (transitivity). $\langle J, I'\rangle, \langle I', I\rangle \vdash \langle I' \cup J, I\rangle$, if $\mathrm{EX}(J, I') \cap \mathrm{VAL}(I) = \emptyset$.

THEOREM 4.3. *The system $T_3$ is sound and complete for tgd's.*

*Proof.*

*Soundness.* Left to the reader.

*Completeness.* Suppose that $D \models \langle I', I\rangle$. Then there are a chase of $I$ by $D$: $I_0, I_1, \cdots$, an element $I_n$ of this chase, and a valuation that is the identity on $I$ such that $h(I') \subseteq I_n$. We leave it to the reader to show, by induction on $k$, that $D \vdash \langle I_k, I\rangle$.

In particular, it follows that $D \vdash \langle I_n, I \rangle$. Since $h(I') \subseteq I_n$ and $h$ is the identity on $I$, $D \vdash \langle I', I \rangle$ by TD3 and TD6. □

**4.4. tgd's and egd's.** The extension of the systems $T_1$, $T_2$, and $T_3$ to egd's is very similar to the extension of $TT_1$, $TT_2$, and $TT_3$ in § 3.4.

The rules for egd's need essentially no change. We change however the covering rule for the sake of uniformity.

ED0 (triviality). $\vdash \langle (a, a), I \rangle$, if $a \in \text{VAL}(I)$.

ED2 (collapsing). $\langle (a_1, a_2), I \rangle \vdash \langle (h(a_1), h(a_2)), h(I) \rangle$, for any valuation $h$.

ED3 (augmentation). $\langle (a_1, a_1), I \rangle \vdash \langle (a_1, a_2), I \cup J \rangle$.

LEMMA 4.1. *Rules* ED2 *and* ED3 *are sound.*

*Proof.* Both rules follow from rule ED1. □

The translation rule needs no change.

ETD0. (translation). $e \vdash e_1$, $e \vdash e_2$.

LEMMA 4.2. *Let $F$ be any sound and complete formal system for tgd's, and let $F'$ be $F \cup \{\text{ETD0}\}$. If $D$ is a set of tgd's and egd's then $D \vdash_{F'} \langle I', I \rangle$ if and only if $D \models \langle I', I \rangle$.*

*Proof.* Identical to the proof of Lemma 3.4.

The mixed (tgd-egd) analogues of rules TD4, TD5 and TD6 turn out to be minor variants of the transitivity rule.

ETD1 (transitivity). $\langle (a_1, a_2), I \rangle$, $\langle I, J \rangle \vdash \langle (a_1, a_2), J \rangle$, if $a_1, a_2 \in \text{VAL}(J)$.

THEOREM 4.4. *Let $F$ be any sound and complete formal system for tgd's, and let $F'$ be $F \cup \{\text{ED0, ED2, ED3, ETD0, ETD1}\}$. Then $F'$ is a sound and complete system for tgd's and egd's.*

*Proof.*

*Soundness.* We have to show that rule ETD1 is sound. Let $K \in \text{SAT}(\langle (a_1, a_2), I \rangle, \langle I, J \rangle)$, and let $h$ be a valuation on $J$ such that $h(J) \subseteq K$. There is an extension $g$ of $h$ to $I$ such that $g(I) \subseteq K$, so $g(a_1) = g(a_2)$. But $a_1, a_2 \in \text{VAL}(J)$, so $h(a_1) = g(a_1) = g(a_2) = h(a_2)$.

*Completeness.* Suppose that $D \models \langle (a_1, a_2), I \rangle$, $a_1 \neq a_2$. By Theorem 2.3 there are a chase of $I$ by $D^*$, an element $I_n$ of this chase, an egd $\langle (a_3, a_4), J \rangle$ from $D$, and a valuation $h$ such that $h(J) \subseteq I_n$, $h(a_3) = a_1$, and $h(a_4) = a_2$. Now $D^* \models \langle I_n, I \rangle$, so by assumption $D^* \vdash \langle I_n, I \rangle$. By ED2 and ED3, $\langle (a_3, a_4), J \rangle \vdash \langle (a_1, a_2), I_n \rangle$, so by ETD0 and ETD1, $D \vdash \langle (a_1, a_2), I \rangle$. □

By using clause (4) in Theorem 2.3 we can generalize Theorem 3.7 to tgd's.

ETD2: $d, e_1 \vdash e$, if $d$ and $e$ are $A$-egd's, and $d$ is nontrivial.

THEOREM 4.5. *Let $F$ be a sound and complete formal system for tgd's, and let $F'$ be $F \cup \{\text{ED0, ETD0, ETD2}\}$. Then $F'$ is a sound and complete system for tgd's and egd's.*

*Proof.* Identical to the proof of Theorem 3.7. □

**4.5. Discussion.** While the systems $TT_1$, $TT_2$, and $TT_3$ all look completely different each from the other, the systems $T_1$, $T_2$, and $T_3$ are very similar. Indeed, since $T_2$ is the simplest system of the three, one may ask why we have bothered to study also $T_1$ and $T_3$. The reason is that derivations by $T_2$ are highly unnatural backward derivations as discussed in § 3.5. In contrast, $T_1/T_3$ are forward/backward systems which yield natural derivations that correspond to known theorem-proving procedures as is shown in [Va3].

Since the implication problem for tgd's is unsolvable [BV2], [CLM], there can be no recursive bound on the size of derivations of tgd's. Furthermore, it can be shown that a recursive bound on the length of the derivations would lead to a recursive bound on the size of the derivations. Thus, there can be no such bound. Nevertheless,

the size of derivations is a reasonable measure to compare between formal systems. We now show that our formal systems can each simulate the other with only a linear increase in the size of derivations. That is, there is a constant $c$, such that if there is a derivation of size $s$ of $d$ from $D$ by $T_i$, for some $1 \leqq i \leqq 3$, then for every $1 \leqq j \leqq 3$, there is a derivation of $d$ from $D$ whose size is at most $cs$. Thus, we can consider all our formal systems as equally "powerful".

$T_1$ simulates $T_2$. We have to simulate rule TD5, but TD5 is just a special case of TD4.

$T_2$ simulates $T_3$. We have to simulate rules TD0, TD3, TD6 and TD7.

TD0′ is just a special case of TD0. Suppose now that $\langle I' \cup J, I\rangle \vdash_{\text{TD3}} \langle I', I\rangle$. We have

$$\vdash_{\text{TD0}} \langle I', I \cup I' \cup J\rangle, \text{ and}$$

$$\langle I', I \cup I' \cup J\rangle, \langle I' \cup J, I\rangle \vdash_{\text{TD5}} \langle I', I\rangle.$$

Suppose that $\langle h(I'), I\rangle \vdash_{\text{TD6}} \langle I', I\rangle$. I.e., $h$ is the identity on $I$. We have

$$\vdash_{\text{TD0}} \langle I', h(I') \cup I\rangle, \text{ and}$$

$$\langle I', h(I') \cup I\rangle, \langle h(I'), I\rangle \vdash_{\text{TD5}} \langle I', I\rangle.$$

Suppose now that $\langle J, I'\rangle, \langle I', I\rangle \vdash_{\text{TD7}} \langle I' \cup J, I\rangle$. I.e., $\text{EX}(J, I') \cap \text{VAL}(I) = \varnothing$. We have

$$\vdash_{\text{TD0}} \langle I' \cup J, I' \cup I \cup J\rangle,$$

$$\langle J, I'\rangle \vdash_{\text{TD2}} \langle J, I' \cup I\rangle \text{ (because } \text{EX}(J, I') \cap \text{VAL}(I) = \varnothing),$$

$$\langle I' \cup J, I' \cup I \cup J\rangle, \langle J, I' \cup I\rangle \vdash_{\text{TD5}} \langle I' \cup J, I' \cup I\rangle, \text{ and}$$

$$\langle I' \cup J, I' \cup I\rangle, \langle I', I\rangle \vdash_{\text{TD5}} \langle I' \cup J, I\rangle.$$

$T_3$ simulates $T_1$. We have to simulate rules TD0 and TD4.

Suppose now that $\vdash_{\text{TD0}} \langle I', I\rangle$. That is, there is a valuation $h$ which is the identity on $I$ such that $h(I') \subseteq I$. We have

$$\vdash_{\text{TD0}'} \langle I, I\rangle,$$

$$\langle I, I\rangle \vdash_{\text{TD3}} \langle h(I'), I\rangle, \text{ and}$$

$$\langle h(I'), I\rangle \vdash_{\text{TD6}} \langle I', I\rangle.$$

Suppose that $\langle I', I \cup J'\rangle, \langle J', J\rangle \vdash_{\text{TD4}} \langle I', I \cup J\rangle$, i.e., $\text{VAL}(I) \cap \text{VAL}(J') \subseteq \text{VAL}(J)$ and $\text{EX}(I', I \cup J') \cap \text{VAL}(J) = \varnothing$. We have

$$\vdash_{\text{TD0}'} \langle I \cup J, I \cup J\rangle,$$

$$\langle J', J\rangle \vdash_{\text{TD2}} \langle J', I \cup J\rangle \text{ (because } \text{EX}(J', J) \cap \text{VAL}(I) = \varnothing),$$

$$\langle I \cup J, I \cup J\rangle, \langle J', I \cup J\rangle \vdash_{\text{TD7}} \langle I \cup J \cup J', I \cup J\rangle,$$

$$\langle I', I \cup J'\rangle \vdash_{\text{TD2}} \langle I', I \cup J \cup J'\rangle \text{ (because } \text{EX}(I', I \cup J') \cap \text{VAL}(J) = \varnothing),$$

$$\langle I \cup J \cup J', I \cup J\rangle, \langle I', I \cup J \cup J'\rangle \vdash_{\text{TD7}} \langle I \cup J \cup I' \cup J', I \cup J\rangle, \text{ and}$$

$$\langle I \cup J \cup I' \cup J', I \cup J\rangle \vdash_{\text{TD3}} \langle I', I \cup J\rangle.$$

The reader can verify that there is such a constant $c$ as claimed above. Moreover, our systems can each simulate the other with only a linear increase in the length of derivations.

**5. Formal systems for subclasses.** An *embedded multivalued dependency* (emvd) [Fag1] is a tgd $\langle\{w\}, \{w_1, w_2\}\rangle$ such that, for all $A \in U$, if $w_1[A] = w_2[A]$ then $w[A] = w_1[A]$. It is usually written $X \twoheadrightarrow Y|Z$, where $X = \{A: w_1[A] = w_2[A]\}$, $Y = \{A: w[A] = w_1[A]\}$, and $Z = \{A: w[A] = w_2[A]\}$. If $\mathrm{VAL}(w) \subseteq \mathrm{VAL}(\{w_1, w_2\})$ then it is an mvd (defined in § 2.3).

Emvd's have attracted a great deal of interest. It is not known whether the implication problem for this subclass is solvable or not. While many inference rules for emvd's are known ([BV1], [Sc], [TKY1], [TKY2]), no sound and complete formal system for them is known.[5] Thus, attention has shifted to finding minimal classes of dependencies that include the class of emvd's as a proper subclass and for which a sound and complete system can be found. Two such classes are the classes of template and binary dependencies which are studied in the following section.

**5.1. Template and binary dependencies.** A *template dependency* (td) [SU1] is a tgd of the form $\langle\{w\}, I\rangle$. The class of td's contain the class of emvd's and it also contains the class of *embedded join dependencies* of [MMS] and the class of *projected join dependencies* of [YP]. The implication problem for td's is known to be unsolvable [GL], [Va4]. (In fact, it is shown in these papers that even for projected join dependencies the problem is unsolvable). We show now that both systems $T_1$ and $T_2$ are sound and complete for td's.

THEOREM 5.1. *The systems $T_1$ and $T_2$ are sound and complete for td's.*

*Proof.* The systems are obviously sound for td's. To show completeness suppose that $D \models \langle I', I\rangle$, where $I' = \{w\}$. Studying the derivation by $T_1$ constructed in the proof of Theorem 4.1, we see that every dependency in the derivation is either of the form $\langle I', J_k\rangle$ or $\langle K', K\rangle$ from $D$. In any case it is a td. Studying the derivation by $T_2$ constructed in the proof of Theorem 4.2, we see that every dependency in the derivation is either of the form $\langle I', I_k\rangle$ or $\langle K', K\rangle$ from $D$. In any case it is a td. □

Sadri and Ullman [SU1] have independently developed a formal system for td's, which turns out to be the system $T_2$ restricted to td's.

A *binary dependency* (bd) is a tgd $\langle I', I\rangle$, where $|I| \leqq 2$. The class of bd's contains the class of emvd's, and it also contains the class of *subset dependencies* of [SW]. It is not known whether the implication problem for this class is solvable or not. None of our systems is complete for bd's, but we can combine rules TD2 and TD7 and get a sound and complete system for bd's.

Let $T_4$ be the system {TD0′, TD1, TD3, TD6, TD7′}, where TD7′ is:

TD7′ (transitivity). $\langle J', J\rangle, \langle I' \cup J, I\rangle \vdash \langle I' \cup J' \cup J, I\rangle$, if $\mathrm{EX}(J', J) \cap \mathrm{VAL}(I \cup I') = \varnothing$.

THEOREM 5.2. *The system $T_4$ is sound and complete for tgd's and for bd's.*

*Proof.* Left to the reader. □

**5.2. Embedded multivalued dependencies.** The class of emvd's lies in the intersection of the class of td's and the class of bd's.[6] There is however a very important distinction between td's and bd's on one hand and emvd's on the other. For any fixed universe $U$ there are infinitely many nonequivalent bd's or td's, but only a finite number of nonisomorphic emvd's. ($\langle I', I\rangle$ is isomorphic to $\langle J', J\rangle$ if there is a one-to-one valuation $h$ such that $h(I') = J'$ and $h(I) = J$.) Thus, for a fixed universe $U$ there is a finite number of instances of the implication problem and a finite number of possible

[5] In contrast, for mvd's the implication problem is solvable ([Beer]), and a sound and complete formal system does exist ([BFH]).

[6] This intersection is exactly the class of subset dependencies of [SW].

derivations, so the implication problem is solvable and there is a sound and complete formal system. (That does not mean that we can effectively find the implication testing algorithm and the formal system for any given $U$). What we would like to have are *uniform* algorithm and system, i.e., an algorithm and a system which are "good" for any $U$. If one considers only bounded inference rules in the style of [Arm], [BFH] then it is known that there is no uniform sound and complete system for emvd's [PP], [SW]. Nevertheless, it is still possible that we can find a sound and complete system using unbounded rules like rule TTD4 or rule JD5 of [BV1], [Va1]. The following theorem says that finding a uniform implication testing algorithm is equivalent to finding a uniform sound and complete formal system.

THEOREM 5.3. *The implication problem for emvd's is solvable if and only if there is a sound and complete formal system for emvd's.*

*Proof.*

*Only if.* Suppose that the implication problem for emvd's is solvable, and consider the formal system consisting of one inference rule:

$$d_1, \cdots, d_k \vdash d, \quad \text{if } \{d_1, \cdots, d_k\} \models d.$$

Clearly, this formal system is sound and complete for emvd's.

*If.* Suppose that $F$ is a sound and complete formal system for emvd's. Let $D$ and $d$ over a universe $U$ be given. To decide whether $D \models d$ we list every possible sequence of emvd's $d_1, \cdots, d_m$ and check whether it is a derivation of $d$ from $D$ by $F$. Inasmuch as there is a finite number of nonisomorphic emvd's over $U$, this process must terminate. Hence, the implication problem for emvd's is solvable. □

*Remark.* The same argument holds for the subsets dependencies of [SW], for the embedded join dependencies of [MMS], and for projected join dependencies of [YP]. Since the implication problem for projected join dependencies is unsolvable [GL], [Va4], this class can not have a sound and complete formal system. There is, however, a subtle point here. The syntax used here is such that there is no need to specify the universe $U$ explicitly, because it is evident from the syntax. Projected join dependencies, embedded join dependencies, subset dependencies, and embedded multivalued dependencies were all introduced originally in a different syntax, in which the universe is not evident (see for example the syntax described in the beginning of § 5). When we study formal systems for such a syntax, it is crucial to know how the formal system handles that lack of explicitness, because that may affect whether a class of dependencies has or does not have a sound and complete formal system. We refer the reader to [Va4] for a more thorough discussion of this point.

**5.3. Decomposition of ttgd's.** An *embedded join dependency* (ejd) [MMS] is a td $\langle \{w\}, I \rangle$ such that for every $A \in U$ and two distinct tuples $u$ and $v$ in $I$, if $u[A] = v[A]$ then $w[A] = u[A]$. If $|I| = 2$ then it is an emvd, if $\text{VAL}(w) \subseteq \text{VAL}(I)$, then it is a *join dependency* (jd) [ABU], [Riss], and if $\text{VAL}(w) \subseteq \text{VAL}(I)$ and $|I| = 2$ then it is an mvd. In [BV1], [MM], [Va1] it is shown that every jd is equivalent to a set consisting of one ejd and several mvd's. That is, a jd can be "decomposed" into weaker dependencies. In this section we provide a decomposition theorem for ttgd's, which implies the above result as a special case.

Let $\langle w, I \rangle$ be a ttgd. The decomposition is based upon a distinction between two kinds of values in $I$: the values which are repeated in $I$ and those that are nonrepeated in $I$. Let $u \in I$ and $A \in U$. $u[A]$ is repeated in $I$ if there is another tuple $v \in I$ such that $u[A] = v[A]$.

Let REP $(I)$ be the set

$$\{A: \text{for some } u \in I,\ u[A] \text{ is repeated in } I\}.$$

For any tuple $u$ and a set $X \subseteq U$, *let* $u_X$ be a tuple such that $u_X[X] = u[X]$ and $u[A]$ is a new distinct value for all $A \in \bar{X}$. Suppose that $I = \{w_1, \cdots, w_m\}$. With each tuple $w_i$ we associate two sets $Y_i = \{A: w[A] = w_i[A]\}$ and $X_i = Y_i - \text{REP}(I)$, and a ttgd $\langle u_i, I'\rangle$, where $I' = I \cup \{w_{\text{REP}(I)}\}$, $u_i[X_i] = w_i[X_i]$, and $u_i[\bar{X}_i] = w_{\text{REP}(I)}[\bar{X}_i]$. $X_1, \cdots, X_m$ is a partition of $\overline{\text{REP}(I)}$.

*Example* 3. Let $U = ABCDEF$, and let $\langle w, I\rangle$ be the ttgd $\langle w, \{w_1, w_2, w_3\}\rangle$:

| | $A$ | $B$ | $C$ | $D$ | $E$ | $F$ |
|---|---|---|---|---|---|---|
| $w$: | $a0$ | $b0$ | $c0$ | $d0$ | $e0$ | $f0$ |
| $w_1$: | $a0$ | $b1$ | $c1$ | $d0$ | $e1$ | $f1$ |
| $w_2$: | $a1$ | $b0$ | $c1$ | $d1$ | $e0$ | $f2$ |
| $w_3$: | $a1$ | $b1$ | $c0$ | $d2$ | $e2$ | $f0$ |

Now we have REP $(I) = ABC$, $Y_1 = AD$, $X_1 = D$, $Y_2 = BE$, $X_2 = E$, $Y_3 = CF$, and $X_3 = F$. $w_{\text{REP}(I)}$ is the tuple

| $A$ | $B$ | $C$ | $D$ | $E$ | $F$ |
|---|---|---|---|---|---|
| $a0$ | $b0$ | $c0$ | $d3$ | $e3$ | $f3$ |

$u_1$, $u_2$, and $u_3$ are the tuples

| | $A$ | $B$ | $C$ | $D$ | $E$ | $F$ |
|---|---|---|---|---|---|---|
| $u_1$: | $a0$ | $b0$ | $c0$ | $d0$ | $e3$ | $f3$ |
| $u_2$: | $a0$ | $b0$ | $c0$ | $d3$ | $e0$ | $f3$ |
| $u_3$: | $a0$ | $b0$ | $c0$ | $d3$ | $e3$ | $f0$ |

THEOREM 5.4. *Let* $\langle w, I\rangle$ *be a ttgd,* $I = \{w_1, \cdots, w_m\}$, *and let* $D = \{\langle\{w_{\text{REP}(I)}\}, I\rangle, \langle u_1, I'\rangle, \cdots, \langle u_m, I'\rangle\}$. Then $\langle w, I\rangle \models D$ and $D \models \langle w, I\rangle$.

*Proof.* We show that $\langle w, I\rangle \vdash D$ and $D \vdash \langle w, I\rangle$.

$\langle w, I\rangle \vdash D$: $\langle w, I\rangle \vdash \langle\{w_{\text{REP}(I)}\}, I\rangle$ by TD6. Define a valuation $h$ on $I$ such that $h(w_i) = w$ and $h(w_j) = w_{\text{REP}(I)}$ for $j \neq i$. Now $h(w) = u_i$, so by TD1, $\langle w, I\rangle \vdash \langle u_i, \{w, w_{\text{REP}(I)}\}\rangle$, and by TD4, $\langle w, I\rangle \vdash \langle u_i, I'\rangle$.

$D \vdash \langle w, I\rangle$: We define a sequence of tuples $v_0, \cdots, v_m$ as follows. $v_0$ is $w_{\text{REP}(I)}$, $v_{i+1}[X_{i+1}] = w_{i+1}[X_{i+1}]$, and $v_{i+1}[\overline{X_{i+1}}] = v_i[\overline{X_{i+1}}]$. Observe that $v_m = w$. We show by induction on $i$ that $D \vdash \langle I \cup \{v_i\}, I\rangle$.

*Basis* $(i = 0)$. Since $v_0 = w_{\text{REP}(I)}$, and by TD0, $\vdash \langle I, I\rangle$, we have that $\langle\{w_{\text{REP}(I)}\}, I\rangle \vdash \langle I \cup \{v_0\}, I\rangle$ by TD7.

*Induction.* Suppose that $D \vdash \langle I \cup \{v_i\}, I\rangle$. Let $h$ be a valuation such that $h(w_{\text{REP}(I)}[\bigcup_{j=1}^{i} X_j]) = w[\bigcup_{j=1}^{i} X_j]$ and $h$ is the identity elsewhere. Then $h(w_{\text{REP}(I)}) = v_i$ and $h(u_{i+1}) = v_{i+1}$. By TD1, $\langle u_{i+1}, I'\rangle \vdash \langle v_{i+1}, I \cup \{v_i\}\rangle$, and by TD7 and TD3, $\langle I \cup \{v_i\}, I\rangle$, $\langle v_{i+1}, I \cup \{v_i\}\rangle \vdash \langle I \cup \{v_{i+1}\}, I\rangle$.

It follows that $D \vdash \langle I \cup \{w\}, I\rangle$, and by TD3, $D \vdash \langle w, I\rangle$. □

Note that the dependencies in $D$ are "weaker" than $\langle w, I\rangle$ because they are implied by $\langle w, I\rangle$, but do not, in general, imply $\langle w, I\rangle$.

We show now how Theorem 5.5 implies the above mentioned decomposition of jd's. An ejd $\langle\{w\}, \{w_1, \cdots, w_m\}\rangle$ is also written as $*[R_1, \cdots, R_m]$, where $R_i = \{A: w[A] = w_i[A]\}$. Thus, if $\langle w, I\rangle$ is the jd $*[Y_1, \cdots, Y_m]$, then $\langle\{w_{\text{REP}(I)}\}, I\rangle$ is the

ejd $*[Y_1 \cap \text{REP}(I), \cdots, Y_m \cap \text{REP}(I)]$. That the ttgd $\langle u_i, I' \rangle$ is equivalent to an mvd is less trivial.

LEMMA 5.1. $\langle u_i, I' \rangle \models Y_i \cap \text{REP}(I) \twoheadrightarrow Y_i$ and $Y_i \cap \text{REP}(I) \twoheadrightarrow Y_i \models \langle u_i, I' \rangle$.

*Proof.* Consider the ttgd $\langle u_i, \{w_i, w_{\text{REP}(I)}\} \rangle$. We have $\{A: w_i[A] = w_{\text{REP}(I)}[A]\} = Y_i \cap \text{REP}(I)$ and $\{A: u_i[A] = w_i[A]\} = Y_i$, so this ttgd is the mvd $Y_i \cap \text{REP}(I) \twoheadrightarrow Y_i$. We show now that $\langle u_i, I' \rangle \vdash \langle u_i, \{w_i, w_{\text{REP}(I)}\} \rangle$ and $\langle u_i, \{w_i, w_{\text{REP}(I)}\} \rangle \vdash \langle u_i, I' \rangle$.

$\langle u_i, \{w_i, w_{\text{REP}(I)}\} \rangle \vdash \langle u_i, I' \rangle$ by TD2. To show the opposite direction, let $h$ be a valuation such that $h(w_i) = w_i$ and $h(w_j) = w_{\text{REP}(I)}$, for $j \neq i$. (Such $h$ can be defined because $\langle w, I \rangle$ is a jd.) Now $h(u_i) = u_i$ and $h(I') = \{w_i, w_{\text{REP}(I)}\}$, so by TD1, $\langle u_i, I' \rangle \vdash \langle u_i, \{w_i, w_{\text{REP}(I)}\} \rangle$. □

**6. Concluding remarks.** Our model is rather restricted since it assumes that the database consists of one relation,[7] and that different attributes have disjoint underlying domains, the so-called "many-sorted" case. While these assumptions offer theoretical advantages [BBG], [Fag2], they are dubious from a practical point of view. It happens that our formal systems of § 4 can be very easily extended to the general case of many relations and nondisjoint domains by simply adjoining a relation name to each tuple. In view of this we think that claims to the naturalness of the universal many-sorted case that are based on its having a sound and complete formal system are not very convincing.

Another dubious assumption is that a relation can have an infinite set of tuples. Since a database is inherently finite, there is a strong justification to define a relation as a finite set of tuples. We say that a set of dependencies $D$ *finitely implies* a dependency $d$, denoted $D \models_f d$, if $d$ is satisfied by every finite relation which satisfies all dependencies in $D$. Unfortunately, it is easy to see that the set $\{\langle D, d \rangle: D \not\models_f d\}$ is recursively enumerable. It follows that if the set $\{\langle D, d \rangle: D \models_f d\}$ is not recursive, then it is not even recursively enumerable. Since the finite implication problem for tgd's is unsolvable [BV2], [CLM], there can be no sound and complete formal system for finite implication. In contrast, if all tgd's in $D$ are total then $D \models d$ iff $D \models_f d$ [BV2], [CLM]. Thus, our formal systems for ttgd's and egd's in § 3 are systems for finite implications as well as for implication.

In § 5.3 we have observed that if the implication problem for a class of dependencies is solvable then this class has a sound and complete formal system. Even in this case there is still an interest in finding an "elegant" formal system, one which has a small number of simple axioms and (preferably bounded) inference rules. A typical example is the propositional calculus, which is a formal system for the resursive set of tautologies of propositional logic. Likewise, the implication problem for jd's is solvable, but we would like to have an elegant formal system for that class or for a minimal class of ttgd's containing it. Another case of interest is that of implication of mvd's by ttgd's and egd's, which is probably the most general case for which an efficient implication testing algorithm does exist [BV3]. These cases will be dealt with in future papers.

Finally, since our dependencies are equivalent to first-order sentences, it is interesting to know what is the relationship between our formal systems and the known formal systems for first-order logic. It turns out that there is indeed a very strong connection between our systems and the system of resolution and paramodulation [CL]. This connection will be described in a future paper.

[7] This assumption is usually called "the universal relation assumption".

## REFERENCES


[ABU] A. V. AHO, C. BEERI AND J. D. ULLMAN, *The theory of joins in relational databases*, ACM Trans. Database Systems, 4 (1979), pp. 297–314.

[Arm] W. W. ARMSTRONG, *Dependency structure of database relationships*, Proc. IFIP 74, North-Holland, Amsterdam, 1974, pp. 580–583.

[ASU] A. V. AHO, Y. SAGIV AND J. D. ULLMAN, *Equivalence among relational expressions*, this Journal, 8 (1979), pp. 218–246.

[BB] C. BEERI AND P. A. BERNSTEIN, *Computational problems related to the design of normal form relational schemas*, ACM Trans. Database Systems, 4 (1979), pp. 30–59.

[BBG] C. BEERI, P. A. BERNSTEIN AND N. GOODMAN, *A sophisticate's introduction to database normalization theory*, Proc. International Conference on VLDB, Berlin, 1978, pp. 113–124.

[Beer] C. BEERI, *On the membership problem for multivalued dependencies*, ACM Trans. Database Systems, 5 (1980), pp. 241–259.

[Bern] P. A. BERNSTEIN, *Synthesizing third normal form relations from functional dependencies*, ACM Trans. Database Systems, 1 (1976), pp. 277–298.

[BFH] C. BEERI, R. FAGIN AND J. H. HOWARD, *A complete axiomatization for functional and multivalued dependencies in database relations*, Proc. ACM Conference on Management of Data, Toronto, 1977, pp. 47–61.

[Bir] G. BIRKHOFF, *On the structure of abstract algebras*, Proc. Cambridge Phil. Soc., 31 (1935), pp. 433–454.

[BV1] C. BEERI AND M. Y. VARDI, *On the properties of join dependencies*, in Advances in Database Theory, H. Gallaire, J. Minker and J. M. Nicolas, eds., Plenum, New York, 1981, pp. 25–72.

[BV2] ———, *The implication problem for data dependencies*, Proc. 8th ICALP, Acre, Israel, 1981, in Lecture Notes in Computer Science 115, Springer-Verlag, Berlin, 1981, pp. 73–85; also Technical Report, Dept. Computer Science, Hebrew University of Jerusalem, May 1980.

[BV3] ———, *A proof procedure for data dependencies*, Technical Report, Dept. Computer Science, Hebrew University of Jerusalem, Dec. 1980.

[BV4] ———, *On the complexity of testing implications of data dependencies*, Technical Report, Dept. Computer Science, Hebrew University of Jerusalem, Dec. 1980.

[CL] C. L. CHANG AND C. R. T. LEE, *Symbolic Logic and Mechanical Theorem Proving*, Academic Press, New York, 1973.

[CLM] A. K. CHANDRA, H. R. LEWIS AND J. A. MAKOWSKY, *Embedded implicational dependencies and their inference problem*, Proc. 13th ACM Annual Symposium on Theory of Computing, 1981, pp. 342–354; J. Comput. Systems Sci., to appear.

[Codd] E. F. CODD, *Further normalization of the database relational model*, in Database Systems, R. Rustin, ed., Prentice-Hall, 1972, Englewood Cliffs, NJ, pp. 33–64.

[Del] C. DELOBEL, *Semantics of relations and the decomposition process in the relational data model*, ACM Trans. Database Systems, 3 (1978), pp. 201–222.

[Fag1] R. FAGIN, *Multivalued dependencies and a new normal form for relational databases*, ACM Trans. Database Systems, 2 (1977), pp. 262–278.

[Fag2] ———, *Horn clauses and database dependencies*, J. Assoc. Comput. Mach., 29 (1982), pp. 252–285.

[FMUY] R. FAGIN, D. MAIER, J. D. ULLMAN AND M. YANNAKAKIS, *Tools for template dependencies*, this Journal, 12 (1983), pp. 36–59.

[GL] Y. GUREVICH AND H. R. LEWIS, *The inference problem for template dependencies*, Proc. ACM Symposium on Principles of Database Systems, Los Angeles, 1982, pp. 221–229.

[HW] L. HENSCHEN AND L. WOS, *Unit refutation and Horn sets*, J. Assoc. Comput. Mach., 21 (1974), pp. 590–605.

[Ma] D. MAIER, *Minimum covers in the relational database model*, J. Assoc. Comput. Mach., 27 (1980), pp. 664–674.

[MM] D. MAIER AND A. O. MENDELZON, *Generalized mutual dependencies and the decomposition of database relations*, Proc. International Conference on VLDB, Rio de Janiero, 1979, pp. 75–82.

[MMS] D. MAIER, A. O. MENDELZON AND Y. SAGIV, *Testing implications of data dependencies*, ACM Trans. Database Systems, 2 (1977), pp. 201–222.

[Nic] J. M. NICOLAS, *First order logic formalization for functional, multivalued and mutual dependencies*, Proc. ACM SIGMOD Conference on Management of Data, 1978, pp. 40–46.

[Pa] J. PAREDAENS, *A universal formalism to express decompositions, functional dependencies and other constraints in a relational database*, Technical Report, University of Antwerp, 1980; also Theoret. Comput. Sci., 19 (1982), pp. 143–160.

[PJ] J. PAREDAENS AND D. JANSSENS, *Decompositions of relations—a comprehensive approach*, in Advances in Database Theory, H. Gallaire, J. Minker and J. M. Nicolas, eds., Plenum, New York, 1981, pp. 73–100.

[PP] D. S. PARKER, JR. AND K. PARSAYE-GHOMI, *Inferences involving embedded multivalued dependencies*, Proc. ACM SIGMOD Conference on Management of Data, Los Angeles, 1980, pp. 52–57.

[Riss] J. RISSANEN, *Theory of relations for databases—a tutorial survey*, Proc. 7th Symp. on Mathematical Foundations of Computer Science, 1978, Lecture Notes in Computer Science 64, Springer-Verlag, Berlin, pp. 537–551.

[Sc] E. SCIORE, *A complete axiomatization of full join dependencies*, J. Assoc. Comput. Mach., 29 (1982), pp. 373–393.

[Sel] A. SELMAN, *Completeness of calculi for axiomatically defined classes of algebras*, Algebra Universalis, 2 (1972), pp. 20–32.

[SU1] F. SADRI AND J. D. ULLMAN, *Template dependencies—A large class of dependencies in relational databases and its complete axiomatization*, J. Assoc. Comput. Mach., 29 (1982), pp. 363–372.

[SU2] ———, *The theory of functional and template dependencies*, Theoret. Comput. Sci., 17 (1982), pp. 317–332.

[SW] Y. SAGIV AND S. WALECKA, *Subset dependencies and a completeness result for a subclass of embedded multivalued dependencies*, J. Assoc. Comput. Mach., 29 (1982), pp. 103–117.

[TKY1] K. TANAKA, Y. KAMBAYASHI AND S. YAJIMA, *On the representability of decompositional schema design with multivalued dependencies*, Technical Report, Kyoto University, 1979.

[TKY2] ———, *Properties of embedded multivalued dependencies in relational databases*, Trans. IECE of Japan, E62, 1979.

[Va1] M. Y. VARDI, *Axiomatization of functional and join dependencies in the relational model*, M.Sc. Thesis, Weizmann Institute of Science, Rehovot, Israel, 1980.

[Va2] ———, *Inferring multivalued dependencies from functional and join dependencies*, Technical Report, Dept. Applied Science, Weizmann Institute of Science, Rehovot, Israel, 1980; Acta Informatica, to appear.

[Va3] ———, *The implication problem for data dependencies in relational databases*, Ph.D. thesis, Hebrew University of Jerusalem, 1981.

[Va4] ———, *The implication and the finite implication problem for typed template dependencies*, Proc. ACM Symposium on Principles of Database Systems, Los Angeles, 1982, pp. 230–238, revised, Technical Report STAN-CS-82-912, Dept. Computer Science, Stanford University, Stanford, CA, May 1982; J. Comput. Systems Sci., to appear.

[YP] M. YANNAKAKIS AND C. PAPADIMITRIOU, *Algebraic dependencies*, 21st IEEE Ann. Symp. on Found. of Computer Science, 1980, pp. 328–332; J. Comput. Systems Sci., 21 (1982), pp. 2–41,

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