Testing Implications of Data Dependencies

DAVID MAIER
State University of New York at Stony Brook
ALBERTO O. MENDELZON
Princeton University
and
YEHOSHUA SAGIV
University of Illinois

Presented is a computation method—the *chase*—for testing implication of data dependencies by a set of data dependencies. The chase operates on tableaux similar to those of Aho, Sagiv, and Ullman. The chase includes previous tableau computation methods as special cases. By interpreting tableaux alternately as mappings or as templates for relations, it is possible to test implication of join dependencies (including multivalued dependencies) and functional dependencies by a set of dependencies.

Key Words and Phrases: data dependencies, join dependencies, multivalued dependencies, functional dependencies, tableaux, chase, relational databases
CR Categories: 4.33, 5.21

1. INTRODUCTION

In the theory of relational databases, the family of integrity constraints known as *data dependencies* plays an important role. Various types of such dependencies have been studied in the literature: functional [3, 11], multivalued [13, 20], and join dependencies [18].

Given a set of dependencies, there are additional dependencies *implied* by this set in the sense that any relation that satisfies the original set must also satisfy the additional dependencies. We often want to know if one dependency is implied by a given set of dependencies. For example, the problem arises in the synthesis

This research was supported in part by the National Science Foundation under Grant MCS 76-15255. The work of A.O. Mendelzon was supported by an IBM Fellowship and the work of Y. Sagiv was supported by a grant from Bell Laboratories.
A version of the work reported herein was presented at the 1979 Conference on Management of Data (SIGMOD), Boston, Mass., June 1979.
Authors' present addresses: D. Maier, Department of Computer Science, State University of New York at Stony Brook, Stony Brook, NY 11794; A.O. Mendelzon, IBM Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, NY 10598; Y. Sagiv, Department of Computer Science, University of Illinois, Urbana IL 61820.

approach to database design [4, 5, 9], in the decomposition approach [e.g., 15], in determining whether relation schemes are normalized [14], in testing for equivalence of database schemes [8], and in determining whether a decomposition satisfies independence properties [16, 17]. The lossless join algorithm of [1] is also an implication test for dependencies.

We shall use tableaux and an operation on tableaux, the *chase*, to determine such consequences of a set of functional and join dependencies. The chase computation may require exponential time in some cases. However, this technique unifies the treatment of functional, multivalued, and join dependencies and provides better insight into the problem. Hence, it may be instrumental in finding new special cases (in addition to those mentioned above) for which polynomial time algorithms exist.

In particular, we show that the chase computation provides a method not only for testing implications, but also for inferring functional and multivalued dependencies. Given a set of dependencies C and a set of attributes X, we can find in no more than exponential time the closure of X and the dependency basis of X.

2. BASIC DEFINITIONS

A *universe* U is a finite set of *attributes* $\{A_1, A_2, \ldots, A_m\}$ and an associated domain D_i for each attribute A_i. Each domain is a countably infinite set. Each *universal element* of U is a mapping $\mu: \{A_1, \ldots, A_m\} \rightarrow \mathbf{D}$, where $\mathbf{D}$ is the union of the D_i's. The mapping must take each attribute to a member of its corresponding domain. If we assign an order to the attributes, then a universal element of U corresponds to a member of the Cartesian product $D_1 \times D_2 \times \cdots \times D_m$. A *universal instance* (or just an *instance*) is a finite set of universal elements, corresponding to a finite subset of this Cartesian product.

A *relation scheme* R over U is a subset of the set of attributes for U. A *relation* r on R is a finite set of mappings or *tuples*, each taking the attributes in R into their corresponding domains. A relation can thus be regarded as a finite subset of the Cartesian product of domains corresponding to attributes in R. Both instances and relations can be viewed as tables with columns corresponding to attributes and rows containing a member from the domain of each attribute.

An instance is really just a relation over the attributes in U. For the moment, a *database scheme* on universe U is a set $\mathbf{R}$ of relation schemes $\{R_1, R_2, \ldots, R_p\}$, where $\bigcup_{i=1}^{p} R_i = U$, and a *database* is a set of relations $\{r_1, r_2, \ldots, r_p\}$, where r_i is a relation on scheme R_i. When dealing with sets of attributes, uppercase letters near the beginning of the alphabet denote single attributes, while uppercase letters from the end of the alphabet denote sets of attributes. Concatenation of sets of attributes denotes union.

Two useful operations on relations are projection and (natural) join. Let $R = XY$ be a relation scheme. The *projection* of a relation r on R onto Y is a relation r' on Y. The relation r' is obtained by removing columns of r not corresponding to attributes in Y and eliminating duplicate tuples in what remains. This projection is denoted $\pi_Y(r)$. Let r be a relation on $R = XY$ and s be a relation on $S = YZ$, with Y the intersection of XY and YZ. The *join* of r and s, denoted $r \bowtie s$, is a relation r' on XYZ. We put tuple t into r' if the XY portion of t is a tuple of r and the YZ portion is a tuple of s. The join is an associative operation, so we may write a string of joins without parenthesizing. We may use projection and join on universal instances by treating the instances as relations on the attributes of U.

The *universal instance assumption* [1, 6] restricts every relation r_i in a database to be the projection of some common universal instance I onto the corresponding schemes R_i. In our notation for projection, there exists a universal instance I such that $r_i = \pi_{R_i}(I)$, $1 \le i \le p$. These projection mappings allow us to go from universal instances to databases. The join operator allows us to go from databases to instances, provided every attribute of the universe appears in some R_i. In this case, $r_1 \bowtie r_2 \bowtie \cdots \bowtie r_p$ is a universal instance, although not necessarily the one from which the r_i's were projected.

Given a set $\mathbf{S}$ of relation schemes, $\{S_1, \ldots, S_q\}$, the *project-join mapping associated with* $\mathbf{S}$, written $M_{\mathbf{S}}$, is defined by

$$M_{\mathbf{S}}(I) = \pi_{S_1}(I) \bowtie \pi_{S_2}(I) \bowtie \cdots \bowtie \pi_{S_q}(I).$$

What we seek, to insure faithful representation of our instances, is a database scheme $\mathbf{R}$ where $M_{\mathbf{R}}(I) = I$ for all universal instances I. As our definitions now stand, there are only trivial database schemes $\mathbf{R}$ with this property. However, in any given application it is unlikely that the set of instances that might ever need to be represented will constitute the entire set of universal instances. Instead, only some subset $\mathbf{P}$ of universal instances will ever have to be considered. This set $\mathbf{P}$ will usually be defined by a set C of constraints on the set of possible instances. For a set of constraints C, let $\mathrm{SAT}(C) = \{I \mid I \text{ satisfies } C\}$. Say an instance I is *C-admissible* (or simply *admissible*, when C is understood) if I is a member of $\mathrm{SAT}(C)$. If c is a single constraint, we write $\mathrm{SAT}(c)$ for $\mathrm{SAT}(\{c\})$.

One class of constraints is the various *data dependencies*, such as functional dependencies (FDs) and multivalued dependencies (MVDs). These dependencies are extensively treated elsewhere [3, 7, 13, 20]. Rissanen introduced another type of data dependency, the join dependency [18]. Let r be a relation on R and $\mathbf{S} = \{S_1, \ldots, S_q\}$ be a set of subsets of R, with the union of the S_i's being R. We say *r joins losslessly* on $\mathbf{S}$ if $r = \pi_{S_1}(r) \bowtie \cdots \bowtie \pi_{S_q}(r)$. If r joins losslessly on $\mathbf{S}$, then we say r satisfies the *join dependency* (JD) $*[S_1, \ldots, S_q]$, which we sometimes write as $*[\mathbf{S}]$. An MVD is a special case of a JD. The MVD $X \to\to Y$ for a relation on R is the JD $*[XY, X(R - Y)]$.

A set of constraints C *implies* a constraint c, written $C \models c$, if $\mathrm{SAT}(c)$ contains $\mathrm{SAT}(C)$. In other words, every instance that satisfies all the constraints of C must also satisfy c. Much work has been done on implications of a set of data dependencies. Following Rissanen [18], let F, M, and J be sets of FDs, MVDs, and JDs, respectively. Let $\mathbf{F}(C)$, $\mathbf{M}(C)$, and $\mathbf{J}(C)$ be the sets of FDs, MVDs, and JDs implied by a set of constraints C. Previous work has dealt with methods for computing $\mathbf{F}(F)$ [3, 9], $\mathbf{M}(F \cup M)$ and $\mathbf{F}(F \cup M)$ [4, 7, 15, 19], $\mathbf{J}(F)$ and $\mathbf{J}(M)$ [1], and $\mathbf{J}(F \cup M)$ [18]. In the sequel, we shall investigate the sets $\mathbf{F}(F \cup M \cup J)$ and $\mathbf{J}(F \cup M \cup J)$; the latter also determines $\mathbf{M}(F \cup M \cup J)$, since every MVD is a JD. Our method therefore handles all these previous situations as special cases.

3. TABLEAUX AND TRANSFORMATION RULES

Tableaux were defined by Aho, Sagiv, and Ullman [2]. We use a simplified definition here, similar to the one of Aho, Beeri, and Ullman [1]. A *tableau* is a set of rows, best pictured as a matrix with one column for each attribute in the universe U. The rows of the matrix are composed of *distinguished variables*, denoted by subscripted a's, and *nondistinguished variables*, denoted by subscripted b's. Each variable may appear in only one column. Furthermore, only one distinguished variable may appear in each column. By convention, the distinguished variable a_i will be the one that appears in the column corresponding to attribute A_i. In this paper we assume that every distinguished variable appears at least once. Below is an example of a tableau for a universe with attributes A, B, and C.

A	B	C
a_1	b_1	a_3
a_1	a_2	b_2
b_3	b_1	b_4

Let T be a tableau and let V be the set of all variables appearing in T. A *valuation* ρ for T is a mapping from V to $\mathbf{D}$, such that $\rho(v)$ is in D_i if v is in the column corresponding to A_i. (Recall that $\mathbf{D}$ is the union of all the domains D_i.) We extend valuations to apply to rows of T in the obvious manner: if w is the row $\langle v_1 v_2 \cdots v_n \rangle$, then $\rho(w)$ is the row $\langle \rho(v_1) \cdots \rho(v_n) \rangle$. We may interpret tableau T as a mapping from instances to instances as follows. Let $w_1, w_2, \ldots, w_n$ be the rows of T. Tableau T maps instance I to $T(I)$, where

$$T(I) = \{\rho(\langle a_1, a_2, \ldots, a_n \rangle) \mid \rho \text{ is a valuation for } T \text{ and } \rho(w_i) \in I, 1 \le i \le n\}.$$

It is important to note that a tableau defined on universe U can always be transformed into an instance of U by applying a valuation to it. In particular, we shall often speak of "tableau $\mathbf{T}$ considered as an instance"; what we mean by this is an instance $I = \sigma(T)$ that results from applying a one-to-one valuation σ to each row of T. For the sake of brevity we shall omit explicit mention of σ in the sequel.

Following Aho et al. [1, 2], we can construct a tableau T representing the mapping $M_{\mathbf{S}}$ for any set $\mathbf{S} = \{S_1, \ldots, S_q\}$ such that the union of the S_i's yields all the attributes in U. T has a row w_i for each S_i. Row w_i has a_j in the jth column if S_i contains A_j. All other entries in the tableau are distinct nondistinguished symbols. In Figure 1 we have the tableau for $M_{\mathbf{S}}$, when $\mathbf{S} = \{ABD, BCE, DE\}$.

Let $\mathbf{P}$ be a set of instances. We say tableaux T_1 and T_2 are *equivalent* on $\mathbf{P}$, written $T_1 \equiv_{\mathbf{P}} T_2$, if $T_1(I) = T_2(I)$ for all I in $\mathbf{P}$. When $\mathbf{P}$ is the set of all instances, we write $T_1 \equiv T_2$ instead of $T_1 \equiv_{\mathbf{P}} T_2$.

A	B	C	D	E
a_1	a_2	b_1	a_4	b_2
b_3	a_2	a_3	b_4	a_5
b_5	b_6	b_7	a_4	a_5

Fig. 1. Tableau for $\{ABD, BCE, DE\}$

The following lemma generalizes a result of [2].

LEMMA 1. *Let T_1 and T_2 be tableaux, and let* $\mathbf{P}$ *be a set of instances. Suppose that there are tableaux $T_1{}^*$ and $T_2{}^*$ such that*

(*a*) $T_1 \equiv_{\mathbf{P}} T_1{}^*$, $T_2 \equiv_{\mathbf{P}} T_2{}^*$, *and*
(*b*) $T_1{}^*$ *and* $T_2{}^*$, *considered as instances, are both in* $\mathbf{P}$.

Then $T_1 \equiv_{\mathbf{P}} T_2$ *if and only if* $T_1{}^* \equiv T_2{}^*$.

PROOF. The if part is immediate. For the other part, suppose $T_1 \equiv_{\mathbf{P}} T_2$. Then $T_1{}^* \equiv_{\mathbf{P}} T_2{}^*$. We need to show $T_1{}^*$ and $T_2{}^*$ are equivalent everywhere. Consider $T_1{}^*(T_1{}^*)$ (note that we are treating $T_1{}^*$ simultaneously as a mapping and as an instance). Since $T_1{}^*$ is an instance in $\mathbf{P}$, it must be that $T_1{}^*(T_1{}^*) = T_2{}^*(T_1{}^*)$. Let w be the row of all distinguished variables. If we choose ρ to be the identity valuation, then $\rho(w_i) = w_i$ is in $T_1{}^*$ for all w_i in $T_1{}^*$ and hence $\rho(w) = w$ is in $T_1{}^*(T_1{}^*)$. Therefore w is in $T_2{}^*(T_1{}^*)$, and there must exist a σ with $\sigma(w) = w$ and $\sigma(w_j)$ in $T_1{}^*$ for all w_j in T_2. Now let I be any instance. Choose any tuple t in $T_1{}^*(I)$. Let ρ be the valuation with $\rho(w) = t$ and $\rho(w_i)$ in I for all w_i in $T_1{}^*$. Consider $\rho\sigma$. It follows that $\rho\sigma(w) = \rho(w) = t$, and $\rho\sigma(w_j)$ is in I for all w_j in $T_2{}^*$. Hence $T_1{}^*(I) \subseteq T_2{}^*(I)$. A symmetric argument will show that $T_1{}^*(I) = T_2{}^*(I)$. □

We now consider methods for modifying tableaux while preserving equivalence. A *transformation rule* for $\mathbf{P}$ is a method for changing a tableau T to a tableau T', with $T \equiv_{\mathbf{P}} T'$. When $\mathbf{P}$ is the set of all instances, the set of possible transformation rules is very limited. When the set of admissible instances is restricted, however, more rules are available. In general, for any constraint c there is a set of transformation rules preserving equivalence on SAT(c). Such rules are essentially a means to incorporate information about the set of admissible instances into the tableau. We present transformation rules for FDs and JDs (and hence MVDs). We shall show that repeated applications of the rules corresponding to a set C of FDs and JDs always yield a tableau to which no further applications of rules can be made. We shall see that this final tableau provides useful information on how the mapping represented by the original tableau behaves on SAT(C). The rules are as follows.

F-Rules. For each FD $X \rightarrow A$, where A is a single attribute, there is a corresponding F-rule. Suppose tableau T has rows w_1 and w_2 that agree in all the X-columns. Let v_1 and v_2 be the variables in the A column of w_1 and w_2, respectively, and suppose that $v_1 \neq v_2$. Applying the F-rule corresponding to $X \rightarrow A$ to rows w_1 and w_2 of T yields a transformed tableau T'. Tableau T' is T with all occurrences of variables v_1 and v_2 identified and duplicate rows removed. The variables are identified by the following renaming rule. If one of the variables is distinguished, the other one is renamed to that distinguished variable. If both are nondistinguished, rename the variable with the larger subscript to be the variable with the smaller subscript.

J-Rules. Let $\mathbf{S} = \{S_1, \ldots, S_q\}$, with the union of the S_i's yielding all the attributes of U. Rows $w_1, \ldots, w_q$ of T (not necessarily distinct) are *joinable* on

A	B	C	D	E
a_1	a_2	a_3	a_4	b_2
b_3	a_2	a_3	b_4	a_5
b_5	b_6	b_7	a_4	a_5

Fig. 2. Result of applying $B \to C$ to tableau of Figure 1

A	B	C	D	E
a_1	a_2	a_3	a_4	b_2
b_3	a_2	a_3	b_4	a_5
b_5	b_6	b_7	a_4	a_5
b_3	a_2	b_7	a_4	a_5

Fig. 3. Result of applying $*[CDE, ABE]$ to tableau of Figure 2

S if there exists a row w not in T that agrees with w_i on S_i, $1 \leq i \leq q$. Row w is the *result* of joining the w_i's. The J-rule corresponding to the JD $*[\mathbf{S}]$ takes rows $w_1, \ldots, w_q$ of T that are joinable on **S** and adds their result w to T to form tableau T'.

Figure 2 shows the result of applying the F-rule for $B \to C$ to rows 1 and 2 of the tableau in Figure 1. Figure 3 is the result of applying $*[CDE, ABE]$ to rows 3 and 2 of Figure 2. We shall sometimes speak of applying an FD or JD to a tableau, meaning the corresponding F-rule or J-rule. Further on we shall apply the term joinable to tuples of an instance. Note that if tuples $u_1, \ldots, u_q$ of instance I are joinable on **S** with result u, and I is in $\text{SAT}(*[\mathbf{S}])$, then u must be in I.

THEOREM 1. *Let T' be the result of applying an F-rule for $X \to A$ to T. Then T and T' are equivalent on* $\text{SAT}(X \to A)$.

PROOF. See Aho, Sagiv, and Ullman [2]. □

THEOREM 2. *Let T' be the result of applying a J-rule for $*[\mathbf{S}]$ to T. Then T and T' are equivalent on* $\text{SAT}(*[\mathbf{S}])$.

PROOF. We must show that $T'(I) = T(I)$ for all I in $\text{SAT}(*[\mathbf{S}])$.

$[T'(I) \subseteq T(I)]$. Let t be a tuple of $T'(I)$. There is some valuation ρ such that $\rho(\langle a_1 \cdots a_m \rangle) = t$ and $\rho(w)$ is in I for every row w in T'. But then t is in $T(I)$, since ρ maps every row in T into something in I, because every row of T is in T'.

$[T(I) \subseteq T'(I)]$. Let t be a tuple of $T(I)$. Let ρ be the valuation that generated t and let w' be the row in T' but not in T. We must show that $\rho(w')$ is in I. Assume w' was formed by a J-rule from rows $w_1, \ldots, w_q$ of T. Hence $w_1, \ldots, w_q$ are joinable on **S** with result w'. We know $\rho(w_1), \ldots, \rho(w_q)$ are all in I and it is not hard to show they are joinable on **S** with result $\rho(w')$. Since I is in $\text{SAT}(*[\mathbf{S}])$, $\rho(w')$ must be in I. Therefore, $t = \rho(\langle a_1, \ldots, a_m \rangle)$ is in $T'(I)$, since $\rho(w)$ is in I for every row w of T'. □

Convention. In the sequel, C will always be a set of FDs and JDs.

4. THE CHASE

In this section we shall show that, when the set of instances **P** is defined by a set of dependencies C (i.e., $\mathbf{P} = \text{SAT}(C)$), the F-rules and J-rules can be used to produce for each tableau T a tableau $\text{CHASE}_C(T)$ from which it is easy to determine whether T is the identity mapping on $\text{SAT}(C)$.

As we shall see, the F-rules and J-rules associated with a set of dependencies C are a *Finite Church-Rosser* (FCR) system. That is, given a tableau T, they can be applied to T only a finite number of times, and the resulting tableau is unique, independently of the order in which the rules were applied. A tableau T' is the *chase of T under C*, written $\text{CHASE}_C(T)$, if it is obtained from T by repeated applications of the rules associated with C, and no rule can be further applied to T'. A *generating sequence* for T is a sequence of tableaux $T_0, T_1, \ldots, T_n$ such that $T_0 = T$, T_i is obtained from T_{i-1} by an application of a rule, and $T_n = \text{CHASE}_C(T)$, i.e., no rule can be applied to T_n. We shall show later (in Lemmas 3 and 4) that a finite generating sequence exists for every tableau T and every set of dependencies C.

Suppose that a tableau T_i is obtained from T_{i-1}. For each row w of T_{i-1} we define its *corresponding row* in T_i as follows. If T_i was obtained by applying a J-rule, then there must be a row v in T_i such that v is identical to w, and we let v be the corresponding row for w. If T_i was obtained by applying an F-rule, then either w appears in T_i or w has been changed by the F-rule to some v, and v appears in T_i. In the first case, w has an identical corresponding row in T_i; in the second case, row v of T_i corresponds to row w of T_{i-1}. Note that two rows of T_{i-1} may have the same corresponding row in T_i.

Let T be a tableau with rows $w_1, \ldots, w_m$, and let $T_0, \ldots, T_n$ be a generating sequence for T under a set of dependencies C. We extend the relation "v corresponds to w" to its transitive-reflexive closure. Thus, for all tableaux T_i in the sequence, there are rows $w_1{}^i, w_2{}^i, \ldots, w_m{}^i$ (not necessarily distinct) in T_i that correspond, respectively, to rows $w_1, \ldots, w_m$ of T.

LEMMA 2. *Let I be an instance in $\text{SAT}(C)$, and let ρ be a valuation of T such that for all rows w_i of T, $\rho(w_i)$ is in I. Then for all tableaux T_i in a generating sequence for T,*

(1) *$\rho(T_i) \subseteq I$, and*
(2) *for all $1 \le j \le m$, $\rho(w_j) = \rho(w_j{}^i)$, i.e., ρ maps corresponding rows of T and T_i to the same tuple in I.*

PROOF. The proof is by induction on i. The basis, for $i = 0$, is immediate. For the induction step, let $i > 0$, and assume the result is true for T_{i-1}. If T_i is obtained from T_{i-1} by an application of a J-rule, then every row of T_{i-1} has an identical corresponding row in T_i, and hence part (2) of our claim is true. Furthermore, we can show as in the proof of Theorem 2 that the new row of T_i is mapped into I by ρ, thus proving part (1). Now suppose T_i is obtained from T_{i-1} by an application of an F-rule, say $X \to A$. Let v_1 and v_2 be the variables in the A-column of rows r_1 and r_2, respectively. Since r_1 and r_2 agree on the X-columns, so do $\rho(r_1)$ and $\rho(r_2)$. Since $\rho(r_1)$ and $\rho(r_2)$ are tuples of I, they must also agree on the A-column. Hence, $\rho(v_1) = \rho(v_2)$. It now follows that for every row w of T_{i-1}, its corresponding row w' in T_i is such that $\rho(w) = \rho(w')$. Since every row of T_i corresponds to some row of T_{i-1}, our claim is true for T_i. □

We are now ready to prove that our system of transformations is FCR.

LEMMA 3. *A given set of F-rules and J-rules can be applied to a tableau T only a finite number of times.*

PROOF. Since tableaux are sets of rows, and none of the rules can introduce new variables, it suffices to show that a tableau cannot appear in a generating sequence more than once. Let T_i and T_j, $i < j$, be two tableaux in a generating sequence for T. If only J-rules were applied to go from T_i to T_j, then T_j contains some row that is not in T_i. If some F-rule was used, then T_i contains some variable that is not in T_j. Hence, T_i and T_j are distinct. □

LEMMA 4. *$\text{CHASE}_C(T) = T$ for tableau T if and only if T, considered as an instance, is in* SAT(C).

PROOF. Suppose T is not in SAT(C). If T violates an FD $X \rightarrow A$, there must be two rows in T that agree on X but not on A. Thus the F-rule for $X \rightarrow A$ can be applied to T yielding a different tableau, and hence $T \neq \text{CHASE}_C(T)$. If T violates a JD of C, the argument is similar. The other implication follows from the fact that a transformation rule for C can be applied to a tableau T only when T, considered as an instance, violates some dependency in C. □

LEMMA 5. *Let $T_0^1, \ldots, T_n^1$ and $T_0^2, \ldots, T_m^2$ be two generating sequences for a tableau T under a set of dependencies C. Then T_n^1 and T_m^2 are identical.*

PROOF. We shall first prove that T_n^1 and T_m^2 are the same up to renaming of nondistinguished variables. This result is valid even if whenever two nondistinguished variables are identified by an F-rule, we choose one of them arbitrarily to replace both variables, instead of following the rule of the smaller subscript as originally defined. If we do follow the rule of the smaller subscript, we shall see that T_n^1 and T_m^2 are identical.

Let ρ_1 and ρ_2 be one-to-one valuations that map T_n^1 and T_m^2 into instances I_1 and I_2, respectively. Note that I_1 and I_2 must both be in SAT(C) by Lemma 4. Let $w_1, \ldots, w_k$ be the rows of T; $w_1^1, \ldots, w_k^1$ are the corresponding rows of T_n^1, and $w_1^2, \ldots, w_k^2$ are the corresponding rows of T_m^2.

Since two rows of T agree on a column only if the corresponding rows of T_n^1 agree on the same column, it is easy to see that there exists a valuation β_1 of T such that for all rows w_i of T, $\beta_1(w_i) = \rho_1(w_i^1)$. By an application of Lemma 2, it follows that (1) $\beta_1(T_m^2) \subseteq I_1$, and (2) $\beta_1(w_i^2) = \beta_1(w_i) = \rho_1(w_i^1)$ for $1 \le i \le k$.

Consider now the mapping $\delta_1 = \rho_1^{-1}\beta_1$. Under δ_1, each variable of T is mapped to a variable of T_n^1. It follows from (1) that δ_1 maps rows of T_m^2 into rows of T_n^1, and from (2) that δ_1 maps every w_i and every w_i^2 into w_i^1. Similarly, we can find a mapping δ_2 from the variables of T to the variables of T_m^2, such that δ_2 maps rows of T_n^1 into rows of T_m^2, and δ_2 maps every w_i and every w_i^1 into w_i^2. It follows that δ_1 maps every distinguished variable of T to itself, since a distinguished variable is never replaced with another variable when going from T to any T_i^1. Furthermore, δ_1 maps different variables of T_m^2 into different variables of T_n^1. In proof, suppose two different variables v_1 and v_2 of T_m^2 are mapped into the same variable of T_n^1. We may assume both variables appear in the same column, say column A, and there must exist two rows w_i^2 and w_j^2 of T_m^2 that disagree on that column. But since the composition $\delta_2\delta_1$ maps w_i^2 into w_i^2 and w_j^2 into w_j^2, the images of these two rows under δ_1 must also disagree on column A.

We conclude from this that an application of δ_1 to T_m^2 can be viewed as a renaming of nondistinguished variables mapping T_m^2 into T_n^1, and similarly for

A	B	C	D	E
a_1	a_2	a_3	a_4	a_5
b_3	a_2	a_3	b_4	a_5
b_5	b_6	b_7	a_4	a_5
b_3	a_2	b_7	a_4	a_5

Fig. 4. Result of applying $BD \to E$ to tableau of Figure 3

A	B	C	D	E
a_1	a_2	a_3	a_4	a_5
b_3	a_2	a_3	b_4	a_5
b_5	b_6	a_3	a_4	a_5
b_3	a_2	a_3	a_4	a_5
b_5	b_6	a_3	b_4	a_5
a_1	a_2	a_3	b_4	a_5
b_5	a_2	a_3	b_4	a_5

Fig. 5. Final tableau, obtained by applying $B \to C$ and $*[CDE, ABE]$ to tableau of Figure 4

δ_2. We have thus proved that T_m^2 and T_n^1 are the same up to renaming of nondistinguished variables.

We now show that T_n^1 and T_m^2 are actually identical. We define an equivalence relation E^1 on the variables of T as follows. For two variables v, w, vE^1w holds if and only if v and w are identified by an F-rule in the generating sequence for T_n^1. We define E^2 similarly using the generating sequence for T_m^2. We extend E^1 and E^2 to their reflexive-transitive closure.

We now claim that E^1 and E^2 are the same equivalence relation. In proof, suppose there are variables v, w such that vE^1w holds but vE^2w does not hold. Let w_i and w_j be rows of T such that v appears in the A-column of w_i and w appears in the A-column of w_j. Since vE^1w holds, it can be shown that w_i^1 and w_j^1 agree in the A-column. Similarly, the fact that vE^2w does not hold implies that w_i^2 and w_j^2 disagree in the A-column. But this a contradiction, since δ_1 maps w_i^2 and w_j^2 to w_i^1 and w_j^1, respectively, and δ_2 maps w_i^1 and w_j^1 to w_i^2 and w_j^2, respectively. An analogous argument shows $E^2 \subseteq E^1$.

Let $E = E^1 = E^2$. Let v be any variable of T. Say v appears in column A. In every row of T_n^1 corresponding to some row of T, v will be replaced by the distinguished variable a in column A if aEv holds; otherwise, v will be replaced by the nondistinguished variable with the smallest subscript among all those that are equivalent to v under E. The same is true in every row of T_m^2. It follows that every row of T_n^1 is a row of T_m^2, and vice versa, and hence the two tableaux are identical. □

Figures 2 through 5 show the computation of the chase for the tableau given in Figure 1 under the constraints $\{B \to C, BD \to E, *[CDE, ABE]\}$. Figure 4 shows the application of the F-rule for $BD \to E$ to the tableau of Figure 3, and Figure 5 is the final tableau. The tableau of Figure 5 is obtained from the one in Figure 4 by applying the F-rule for $B \to C$ once and then applying the J-rule for $*[CDE, ABE]$ three times.

We note that the computation for $\text{CHASE}_C(T)$ as given here may take exponential time. We can show that the computation can be done in no more than exponential time as follows. Applying an F-rule to a tableau T can be done in time polynomial in the size of T. Applying a J-rule to a tableau T can be done in time exponential in the number of variables of T. Since the number of tableaux in a generating sequence for T is at most exponential in the number of variables of T, the whole computation takes no more than exponential time in the size of C and T.

The rules can be modified into slightly stronger versions. The F-rule can be extended to apply to FDs $X \rightarrow W$, W a set of attributes. This extended F-rule equates variables in more than one column at once. The J-rule may be strengthened to generate all the tuples allowed by a JD at once. Chasing is still an exponential process, even with these stronger rules. The reason is that one application of a strengthened J-rule can increase the number of rows exponentially. Whether this performance can be improved is an open question. In special cases, such as C consisting only of FDs, the algorithm runs in polynomial time [1].

We have proved in Lemma 5 that the final result of the chase process does not depend on the order in which we apply the rules. The following theorem shows that the result of the chase is also independent of the particular cover of the set of dependencies that we use.

THEOREM 3. *If* $\text{SAT}(C) = \text{SAT}(D)$, *then* $\text{CHASE}_C(T) = \text{CHASE}_D(T)$ *for any tableau* T.

PROOF. We shall first prove a special case of the theorem, where $D = C \cup \{c\}$ for any c such that $C \vDash c$. Let $T' = \text{CHASE}_C(T)$. We can get to T' using rules in D, since $C \subseteq D$. Furthermore, by Lemma 4, no rule for c can be applied to T', since T' viewed as an instance is in $\text{SAT}(C)$ and hence in $\text{SAT}(D)$. So $\text{CHASE}_D(T) = T'$.

Now we drop the restriction on C and D. Note that for any c in C and d in D, $C \vDash d$ and $D \vDash c$. Let $E = C \cup D$. By repeated use of the special case above, we can show $\text{CHASE}_C(T) = \text{CHASE}_E(T) = \text{CHASE}_D(T)$. □

Theorem 3 tells us that before computing the chase under C, we are free to choose any D with $\text{SAT}(C) = \text{SAT}(D)$. In some cases, we may be able to choose a D that will simplify the computation of the chase.

THEOREM 4. *Let T be the tableau corresponding to a project-join mapping $M_{\mathbf{R}}$ and let C be a set of FDs, MVDs, and JDs. Then $M_{\mathbf{R}}$ is the identity on* $\text{SAT}(C)$ *if and only if* $\text{CHASE}_C(T)$ *contains the row of all distinguished variables.*

PROOF. Let $\mathbf{P} = \text{SAT}(C)$. Let T' be a tableau containing only the row of all distinguished variables. Obviously $\text{CHASE}_C(T') = T'$. By Theorems 1 and 2, $T \equiv_{\mathbf{P}} T'$ if and only if $\text{CHASE}_C(T) \equiv_{\mathbf{P}} T'$ (because $T' \equiv_{\mathbf{P}} \text{CHASE}_C(T')$). By Lemmas 1 and 4, $\text{CHASE}_C(T) \equiv_{\mathbf{P}} T'$ if and only if $\text{CHASE}_C(T) \equiv T'$. It follows from the results of [2, 10] that $\text{CHASE}_C(T)$ and T' are equivalent on all instances if and only if $\text{CHASE}_C(T)$ contains the row of all distinguished variables. □

The theorem gives a method of checking whether a set C of FDs, MVDs, and JDs implies any given JD (hence any MVD). Given the JD $*[\mathbf{S}]$, we know instance I satisfies $*[\mathbf{S}]$ if and only if $M_{\mathbf{S}}(I) = I$. The theorem gives a test for $M_{\mathbf{S}}(I) = I$ for all I in $\text{SAT}(C)$. If so, then $\text{SAT}(C) \subseteq \text{SAT}(*[\mathbf{S}])$, that is, $C \vDash *[\mathbf{S}]$.

5. TABLEAUX AS TEMPLATES

We have been interpreting tableaux as mappings from instances to instances. Tableaux may also be considered templates (actually partial templates) for instances. In this section we shall show that this interpretation can be used to find the closure and the dependency basis of a set of attributes X, and also to determine if an FD is implied by a set C.

5.1. Finding the Closure and Dependency Basis

Let C be a set of dependencies, and let $X \rightarrow A$ be a nontrivial functional dependency. We construct a tableau T_X as follows. Tableau T_X has two rows. One row, denoted w_1, has distinguished variables in all the columns. The other row, w_2, has distinguished variables in all the X-columns and nondistinguished variables in the rest of the columns. The following theorem shows how to use T_X to test whether $X \rightarrow A$ is implied by C.

THEOREM 5. *$C \models X \rightarrow A$ if and only if* $\text{CHASE}_C(T_X)$ *has only a distinguished variable in the A-column.*

PROOF. For the only if, let T' be $\text{CHASE}_C(T_X)$ and suppose that T' has more than one variable in the A-column. Then T' serves as a counterexample to the implication, i.e., as an instance in which all the dependencies of C hold, but $X \rightarrow A$ fails. Conversely, suppose that T' has only a distinguished variable in the A-column, and let I be an instance in which C holds. Let t_1 and t_2 be tuples of I that agree in all the X-columns. We can apply the chase computation for T_X to t_1 and t_2. Whenever a new row has to be added by a J-rule, this row is already in I. Whenever two variables are identified, they must already be the same in I. Since eventually the chase computation identifies the variables in the A-column, t_1 and t_2 must agree in that column. Thus, $X \rightarrow A$ holds in I. This shows that $X \rightarrow A$ is implied by C. □

When we want to determine whether $C \models X \rightarrow A$, it suffices to apply the F-rules and J-rules only until column A contains only a distinguished variable, because beyond this point no rule can introduce new variables into the column. Furthermore, in this way we can find the closure of X under C. The *closure of X under C*, or just *closure of X* (denoted X^*) when C is understood, is the set of all attributes A such that $C \models X \rightarrow A$.

COROLLARY 1. *X^* is the set of all attributes A such that the A-column of* $\text{CHASE}_C(T_X)$ *has only distinguished variables.*

COROLLARY 2. $\mathbf{F}(M \cup J) = \{X \rightarrow Y \mid Y \subseteq X\}$. *That is, the only FDs implied by a set of JDs are the trivial ones.*

PROOF. $\text{CHASE}_C(T_X)$, where $C = M \cup J$, will have a nondistinguished variable in every column not associated with an attribute in X, since J-rules cannot identify variables. □

Note that, when C contains only MVDs, the previous corollary follows from the inference rules of Beeri, Fagin, and Howard for FDs and MVDs [7].

A *multivalued dependency* [12, 13, 20] is a join dependency whose associated tableau has no more than two rows. An MVD can also be written as a statement $X \rightarrow\rightarrow Y$, where X and Y are sets of attributes. The corresponding join dependency is $*[XY, X(U - XY)]$. The tableau T for the MVD $X \rightarrow\rightarrow Y$ has one row with distinguished variables exactly in the columns for $X \cup Y$, and a second row with distinguished variables exactly in the columns for $X \cup Z$, where $Z = U - XY$.

Let C be a set of dependencies, and X a set of attributes. Consider the set

$$\Omega = \{Y \mid C \models X \to\to Y\}.$$

Note that the elements of this set are sets of attributes. The inference rules for MVDs [7] imply that Ω has a subset, called the *dependency basis* of X, such that the sets in the dependency basis are pairwise disjoint, every attribute is in some set of the dependency basis, and if $X \to\to Y$ is implied by C, then Y is a union of sets taken from the dependency basis of X. Note that for each attribute A in X^*, $\{A\}$ is a member of the dependency basis of X, since $X \to A \models X \to\to A$.

It is easy to see that the dependency basis of X contains all sets of attributes Y such that (a) $Y \in \Omega$ and (b) Y does not have any proper subset Y' that is also a member of Ω. Thus, if X^* is given and if the set $\Omega' = \{Y \mid C \models X \to\to Y$ and $X^* \cap Y = \phi\}$ is given, then the dependency basis of X can be constructed in time polynomial in the size of X^* and Ω'. We have already seen that X^* can be found in linear time from $\text{CHASE}_C(T_X)$. The following lemma shows that so can Ω'.

Lemma 6. *Let Y be a set of attributes disjoint from X^*. The MVD $X \to\to Y$ is implied by C if and only if $\text{CHASE}_C(T_X)$ contains a row with distinguished variables exactly in the columns for $X^* \cup Y$.*

Proof. (if) Let w_1 be the row of T_X that has only distinguished variables, and let w_2 be the other row of T_X. Let v_1 and v_2 be the corresponding rows in $\text{CHASE}_C(T_X)$. Suppose that row v of $\text{CHASE}_C(T_X)$ has distinguished variables exactly in the columns for $X^* \cup Y$. Let T' be the tableau corresponding to the database scheme $\{XY, XZ\}$, where $Z = U - XY$. Let r_1 be the row of T' corresponding to XY, and r_2 the other row of T', and s_1, s_2 the corresponding rows in $\text{CHASE}_C(T')$.

It is not hard to construct, as in the proof of Lemma 5, a mapping δ from the variables of T_X to the variables of $\text{CHASE}_C(T')$, such that $\delta(w_1) = s_1$ and $\delta(w_2) = s_2$. By an application of Lemma 2, every row of $\text{CHASE}_C(T_X)$ is mapped by δ into a row of $\text{CHASE}_C(T')$. Also, $\delta(v_1) = \delta(w_1)$ and $\delta(v_2) = \delta(w_2)$. It follows that rows s_1 and s_2 must agree in the X^*-columns, since v_1 and v_2 do. But s_1 and s_2 agree on a column only if both have the same distinguished variable in that column. Therefore, δ maps every distinguished variable in the X^*-columns of $\text{CHASE}_C(T_X)$ to a distinguished variable of $\text{CHASE}_C(T')$.

Furthermore, it is easy to see that all distinguished variables of T_X in the Y-columns are mapped by δ to distinguished variables of $\text{CHASE}_C(T')$. Similarly it follows that all nondistinguished variables of T_X in the Z-columns are mapped by δ to distinguished variables of $\text{CHASE}_C(T')$.

Consider now the row v of $\text{CHASE}_C(T_X)$ that has distinguished variables exactly in the columns for $X^* \cup Y$. We claim that the row $\delta(v)$ has only distinguished variables. This follows from the previous remarks and the fact that row v has distinguished variables in the columns for $X^* \cup Y$ and nondistinguished variables in the columns for $Z - X^*$. Therefore, $\delta(v)$ is a row of $\text{CHASE}_C(T')$ with only distinguished variables, proving that $C \models X \to\to Y$.

(Only if) Suppose $C \models X \to\to Y$. By Theorem 4, $\text{CHASE}_C(T_X)$ is the same as $\text{CHASE}_{C \cup \{X \to\to Y\}}(T_X)$. There is a computation of this chase that starts out by applying $X \rightarrow\rightarrow Y$ to rows w_1 and w_2, producing a row w with distinguished variables exactly in the columns for $X \cup Y$. It is easy to see that the row corresponding to w in the final chase under this augmented set of dependencies has distinguished variables exactly in the columns for $X^* \cup Y$. □

COROLLARY 3. *Given a set of dependencies C and a set of attributes X, the closure of X and the dependency basis of X can be found in exponential time.*

5.2. Completions

In this subsection we shall give a more formal setting for the idea of tableaux as templates by defining the notion of a completion of an instance and proving some general results about these completions.

If I is any instance, a *completion* of I under $\mathbf{P}$ is an instance H in $\mathbf{P}$ such that H contains I and there is no proper subset of H in $\mathbf{P}$ containing I. An instance may not always have a completion in $\mathbf{P}$. However, we do have the following result.

LEMMA 7. *Let* $\mathbf{P}$ *be a set of universal instances.* $\mathbf{P}$ *is closed under intersection if and only if completions under* $\mathbf{P}$ *are unique.*

PROOF. Suppose $\mathbf{P}$ is closed under intersection. Let I be an arbitrary instance with completions H and H' under $\mathbf{P}$. Then $H \cap H'$ is in $\mathbf{P}$ and contains I. It follows that $H = H'$. For the converse, suppose completions are unique. Let I and H be in $\mathbf{P}$, and $J = I \cap H$. There must be some subset I' of I that is a completion of J, and some subset H' of H that is a completion of J. But then $I' = H'$, hence $I' = J = H'$, so J is in $\mathbf{P}$. □

Given tableau T and valuation ρ, let $\rho(T)$ be the instance containing $\rho(w)$ for all w in T. We shall view T as a representative of the set of instances

$$\text{REP}(T, \mathbf{P}) = \{I \mid I \text{ is a completion of } \rho(T) \text{ for some valuation } \rho\}.$$

For $\mathbf{P} = \text{SAT}(C)$, we bend our notation and write $\text{REP}(T, C)$ for $\text{REP}(T, \text{SAT}(C))$. It is easy to show that $\text{SAT}(C)$ is closed under intersection.

The following lemma shows how the REP sets of equivalent tableaux are related to each other.

LEMMA 8. *For* $\mathbf{P}$ *closed under intersection, if* $T_1 \equiv_{\mathbf{P}} T_2$*, then for every* I *in* $\text{REP}(T_1, \mathbf{P})$ *there exists an* H *in* $\text{REP}(T_2, \mathbf{P})$ *such that* $H \subseteq I$.

PROOF. Let $I \in \text{REP}(T_1, \mathbf{P})$, where I is the completion of $\rho_1(T_1)$, and let w be the row of all distinguished variables. $T_1(I)$ contains $\rho_1(w)$, since I contains $\rho_1(r)$ for every row r of T_1. Since $T_1 \equiv_{\mathbf{P}} T_2$, $\rho_1(w) \in T_2(I)$. There must be a ρ_2 with $\rho_2(w) = \rho_1(w)$ and $\rho_2(x) \in I$ for every x in T_2. Let the completion of $\rho_2(T_2)$ under $\mathbf{P}$ be H. H exists because $\rho_2(T_2) \subseteq I$. It follows that $H \subseteq I$. □

We think it would be interesting to resolve the following open question: If $\text{REP}(T_1, \mathbf{P}) = \text{REP}(T_2, \mathbf{P})$, and $\mathbf{P}$ is closed under intersection, can it be shown that $T_1 \equiv_{\mathbf{P}} T_2$?

6. FURTHER QUESTIONS

Fagin has introduced the notion of *embedded multivalued dependencies* (EMVDs) [13]. An EMVD takes the form $X \rightarrow\rightarrow Y(Z)$ (read "X multivalued implies Y in the context of Z"). Let $W = XYZ$. An instance I satisfies $X \rightarrow\rightarrow$

A	B	C	D	E
a_1	a_2	b_1	a_4	b_2
b_3	a_2	a_3	b_4	a_5
b_5	b_6	b_7	a_4	a_5
a_1	a_2	a_3	b_8	b_9

Fig. 6. Applying an EMVD rule for $*[AB, BC]$

$Y(Z)$ if its projection onto W satisfies $X \rightarrow\rightarrow Y$. We can similarly define *embedded join dependencies* (EJDs). Let $\mathbf{S} = \{S_1, \ldots, S_q\}$, and let $W = \cup_i S_i$. We do not require that W contain all the attributes in U. The EJD holds in instance I if the projection of I onto W satisfies $*[\mathbf{S}]$.

Can we incorporate EJDs (and hence EMVDs) into C for our chase computation? One possible way to apply the EJD $*[\mathbf{S}]$ on W to a tableau T is to project T onto W, apply the usual J-rule to generate a new row w, pad w to w' with new nondistinguished variables, and add w' to T. Figure 6 shows the result of applying the EJD $*[AB, BC]$ to rows 1 and 2 of the tableau of Figure 1. However, our proof of termination depends on no new variables being added.

Another question is: can we find rules of inference for the set of FDs and JDs similar to those for FDs and MVDs [7]?

ACKNOWLEDGMENTS

The authors wish to thank Catriel Beeri and Jeff Ullman for their comments on an earlier version of this paper.

REFERENCES

1. AHO, A.V., BEERI, C., AND ULLMAN, J.D. The theory of joins in relational databases. Proc. 18th Symp. on Foundations of Computer Science, Providence, R.I., 1977, pp. 107–113.
2. AHO, A.V., SAGIV, Y., AND ULLMAN, J.D. Equivalence of relational expressions. *SIAM J. Comptng. 8*, 2 (May 1979), 218–246.
3. ARMSTRONG, W.W. Dependency structures of data base relationships. Proc. IFIP '74, North-Holland Pub. Co., Amsterdam, 1974, pp. 580–583.
4. BEERI, C. On the membership problem for multivalued dependencies in relational databases. Tech. Rep. 229, Dept. Elec. Eng. and Comptr. Sci., Princeton U., Princeton, N.J., 1977.
5. BEERI, C. On the role of data dependencies in the construction of relational database schemas. Tech. Rep. 43, Dept. Comptr. Sci., The Hebrew University, Jerusalem, Israel, 1979.
6. BEERI, C., BERNSTEIN, P., AND GOODMAN, N. A sophisticate's introduction to database normalization theory. Proc. 4th Int. Conf. on Very Large Data Bases, West Berlin, 1978, pp. 113–124.
7. BEERI, C., FAGIN, R., AND HOWARD, J. A complete axiomatization for functional and multivalued dependencies. Proc. ACM-SIGMOD Conf., Toronto, Canada, 1977, pp. 47–61.
8. BEERI, C., MENDELZON, A.O., SAGIV, Y., AND ULLMAN, J.D. Equivalence of relational database schemes. Proc. 11th ACM Symp. on Theory of Computing, Atlanta, Ga., 1979, pp. 319–329. See also Tech. Rep. 252, Dept. Elec. Eng. and Comptr. Sci., Princeton U., Princeton, N.J., 1978.
9. BERNSTEIN, P.A. Synthesizing third normal form relations from functional dependencies. *ACM Trans. Database Syst. 1*, 4 (Dec. 1976), 277–298.
10. CHANDRA, A.K., AND MERLIN, P.M. Optimal implementation of conjunctive queries in relational databases. Proc. 9th Ann. ACM Symp. on Theory of Computing, Boulder, Colo., 1976, pp. 77–90.
11. CODD, E.F. A relational model for large shared data banks. *Comm. ACM 13*, 6 (June 1970), 377–387.
12. DELOBEL, C. Contributions theoretiques a la conception et a l'evaluation d'un systeme d'informations applique a la gestion. These d'Etat, U. of Grenoble, Grenoble, France, 1973.

13. FAGIN. R. Multivalued dependencies and a new normal form for relational databases. *ACM Trans. Database Syst. 2*, 3 (Sept. 1977), 262-278.
14. FAGIN, R. Normal forms and relational database operators. Proc. ACM-SIGMOD Conf., Boston, Mass., 1979, pp. 153-160.
15. HAGIHARA, K., ITO, M., TANIGUCHI, K., AND KASAMI, T. Decision problems for multivalued dependencies in relational databases. *SIAM J. Comptng. 8*, 2 (May 1979), 247-264.
16. MAIER, D., MENDELZON, A.O., SADRI, F., AND ULLMAN, J.D. Adequacy of decompositions of relational databases. Unpub. manuscript.
17. RISSANEN, J. Independent components of relations. *ACM Trans. Database Syst. 2*, 4 (Dec. 1977), 317-325.
18. RISSANEN, J. Theory of relations for databases—a tutorial survey. Proc. 7th Symp. on Mathematical Foundations of Computer Science, Lecture Notes in Computer Science 64, Springer-Verlag, 1978, pp. 536-551.
19. SAGIV, Y. An algorithm for inferring multivalued dependencies that works also for a subclass of propositional logic. Rep. UIUCDCS-R-79-954, Dept. Comptr. Sci., U. of Illinois, Urbana-Champaign, Ill., 1979.
20. ZANIOLO, C. Analysis and design of relational schemata for database systems. Tech. Rep. UCLA-ENG-7769, Ph.D. Th., Dept. Comptr. Sci., U. of California, Los Angeles, Calif., 1976.

Received March 1979; revised June 1979