STOC(Milwaukee 1981),355-362.

PROPERTIES OF ACYCLIC DATABASE SCHEMES

Catriel Beeri, Hebrew University of Jerusalem†
Ronald Fagin, IBM Research Lab, San Jose
David Maier, SUNY, Stonybrook‡
Alberto Mendelzon, Univ. of Toronto††
Jeffrey Ullman, Stanford Univ.‡‡
Mihalis Yannakakis, Bell Laboratories, Murray Hill

ABSTRACT

There is a class of database descriptions, involving one "acyclic" join dependency and a collection of functional dependencies, and nothing else, that appears powerful enough to describe most any real-world body of data in relational database terms. Further, this class has many desirable properties. Some properties make operations like updates and the selection of joins to implement a query over a universal relation especially easy. Other properties of interest were studied by other researchers who described the same class in radically different terms, and found desirable properties in their own contexts. It is the purpose of this paper to define the class formally, to give its important properties and the equivalences with the other classes mentioned, and to explain the importance of each property. This paper is intended to summarize the results that will appear in more detail in [FMU] and [BFMY].

I. Definitions

A *relational database scheme* consists of a *universal set of attributes* U and a set of "dependencies." The attributes in U are names for the components (columns) of relations in the database, and the dependencies, in the most general sense, are simply subsets of the set of all possible relations over U. The purpose of asserting dependencies is to limit the set of relations that could be the "current value" of the database; by doing so we may be able to use database organizations that we could not use if we knew nothing about what might appear in the database. The most common forms of dependencies are *functional dependencies* (FD's) and *multivalued dependencies* (MVD's).

Functional dependencies are denoted syntactically by $X \rightarrow Y$, where X and Y are (not necessarily disjoint) subsets of U, the set of all attributes. The semantics of FD's is as follows. We say that a relation R satisfies FD $X \rightarrow Y$ if whenever R has two tuples that agree on all the attributes in X, those tuples also agree on the attributes in Y. Thus, $X \rightarrow Y$ defines a subset of all possible relations over U, those relations that satisfy $X \rightarrow Y$.

In database theory, a *tuple* is formally regarded as a mapping from attributes to values, rather than as a list of component values, although the latter viewpoint is handy when the order of the attributes in the list is understood. We often use $t[Z]$, where t is a tuple and Z is a set of attributes, to stand for t restricted to domain Z, that is, the components of t for the attributes in Z. Thus the meaning of FD $X \rightarrow Y$ for a relation R can be stated more precisely as:

if t_1 and t_2 are in R, and $t_1[X] = t_2[X]$,
then $t_1[Y] = t_2[Y]$.

Multivalued dependencies are denoted syntactically by $X \twoheadrightarrow Y$. The meaning of this dependency is that if relation R satisfies the dependency, then for every t_1 and t_2 in R, if $t_1[X] = t_2[X]$ then there exists t_3 in R such that:

1. $t_3[X] = t_1[X] = t_2[X]$,
2. $t_3[Y] = t_1[Y]$, and
3. $t_3[U - X - Y] = t_2[U - X - Y]$.

Less formally, whenever we have two tuples that agree on X, we can interchange their values for the attributes in Y, as a group, and be assured that the two new tuples are also in R.

Example 1: Consider the relation R in Fig. 1, where $U = \{A, B, C, D\}$. The FD $AC \rightarrow D$ holds in R. In proof, note that the first and fourth tuples agree on

† Work performed at IBM Research Lab., San Jose and at Stanford University. Partially supported by NSF grant MCS-79-04528.
‡ Partially supported by NSF grant IST-79-18264.
†† Work performed while at IBM T. J. Watson Research Center, Yorktown Hts., N. Y.
‡‡ Partially supported by Air Force grant AFOSR-80-0212.

both A and C, but they also agree on D. Similarly, the second and third tuples agree on A and C, but also on D. No other pairs of tuples agree on A and C.

A	B	C	D
0	1	2	3
0	2	1	4
0	1	1	4
0	2	2	3
5	1	3	2

Fig. 1. The relation R.

The FD $CD \rightarrow B$ fails to hold, since the second and third tuples agree on C and D, but not on B.

The MVD $A \twoheadrightarrow B$ holds. For example, if t_1 and t_2 are the first two tuples in Fig. 1, then we may check that the tuple t_3, where $t_3[A] = t_1[A] = 0$, $t_3[B] = t_1[B] = 1$, and $t_3[CD] = t_2[CD] = 1,4$, is present; it is row three. []

A *join dependency* is denoted syntactically by

$$\bowtie(R_1, \ldots, R_n)$$

where each R_i is a set of attributes (called an *object*). To describe the meaning of a join dependency, we need to define projection and join. The *projection* of a relation R onto set of attributes $S \subseteq R$, denoted $\pi_S(R)$, is the set of tuples in R restricted to S, that is, $\{ t[S] \mid t \text{ is in } R \}$. For example, the projection of R in Fig. 1 onto set of attributes $\{A, B\}$ is the relation $\{01, 02, 51\}$.

The *join* of relations $R_1, \ldots, R_n$ is the set of tuples over the union of the attributes of $R_1, \ldots, R_n$, such that the projection of the tuple onto each R_i is present in that relation. We can write

$$R_1 \bowtie \cdots \bowtie R_n = \{ t \mid (\forall i) t[R_i] \text{ is in } R_i \}$$

Note that the above purposely confuses the set of attributes over which a relation is defined (the *relation scheme*) with the name of that relation. This convention is useful and, we trust, not really confusing.

Now, we say a relation R satisfies the join dependency $\bowtie(R_1, \ldots, R_n)$ if $R = \pi_{R_1}(R) \bowtie \cdots \bowtie \pi_{R_n}(R)$, that is, when we project R onto the sets of attributes $\{R_i\}$, then join the results, we get back no more than we started with.

Example 2: The relation of Fig. 1 satisfies the join dependency $\bowtie(AB, BC, CD)$, as the reader may check. It does not satisfy $\bowtie(AB, AC, AD)$. For example, the projection onto AB includes the tuple 01, the projection onto AC includes 02, and the projection onto AD includes 04. Thus, the join of the three projections includes 0124, a tuple that is not in R of Fig. 1. []

Note that the multivalued dependency is a special case of the join dependency. That is, the multivalued dependency $X \twoheadrightarrow Y$ in the set of attributes U is equivalent to the join dependency

$$\bowtie(XY, X(U - X - Y))$$

II. Join Dependencies and Hypergraphs

A *hypergraph* is a pair $(\mathcal{N}, \mathcal{E})$, where $\mathcal{N}$ is a set of *nodes*, and $\mathcal{E}$ a set of *edges*, which are arbitrary subsets of $\mathcal{N}$. An ordinary graph is, of course, a hypergraph whose every edge is of size 2.

The *hypergraph of a join dependency*

$$j = \bowtie(R_1, \ldots, R_n)$$

has a set of nodes corresponding to the attributes mentioned in one or more of the R_i's, and a set of edges $\{R_1, \ldots, R_n\}$.

To begin, let us give some terminology for hypergraphs. A *path* from node p to node q is a sequence of $k \geq 1$ edges $E_1, \ldots, E_k$ such that

1. p is in E_1,
2. q is in E_k, and
3. for all $1 \leq i < k$, $E_i \cap E_{i+1}$ is nonempty.

We also say the above sequence of edges is a path from edge E_1 to edge E_k.

Two nodes (or attributes) are *connected* if there is a path from one to the other. Similarly, two edges are *connected* if there is an edge-path from one to the other. A set of nodes or edges is connected if every pair is connected.

Let $(\mathcal{N}, \mathcal{E})$ be a hypergraph. Its *reduction* $(\mathcal{N}, \mathcal{E}')$ is obtained by removing from $\mathcal{E}$ each edge that is a proper subset of another edge. A hypergraph is *reduced* if it equals its reduction, that is if no edge is a subset of another edge. In what follows, we shall assume that a hypergraph is reduced unless stated otherwise. Note that the join dependency of a hypergraph and the join dependency of its reduction are logically equivalent [BMSU].

Let $\mathcal{M}$ be a set of nodes of the hypergraph $(\mathcal{N}, \mathcal{E})$. The set of *partial edges generated by* $\mathcal{M}$ is obtained by intersecting the edges in $\mathcal{E}$ with $\mathcal{M}$, that is

$$\{ (E \cap \mathcal{M}) \mid E \text{ is in } \mathcal{E} \}$$

and then taking the reduction of this set of edges. The set of partial edges generated from $(\mathcal{N}, \mathcal{E})$ by some set $\mathcal{M}$ is said to be a *node-generated set of partial edges*.

Let $\mathcal{F}$ be a connected set of partial edges, and let E and F be in $\mathcal{F}$. Let $Q = E \cap F$. We say that (E, F) is an *articulation pair*, and that Q is an *articulation set* of $\mathcal{F}$, if the result of removing Q from every edge in $\mathcal{F}$ is not a connected set of partial edges. Evidently, an articulation set in a hypergraph is a generalization of the concept of an articulation point in an ordinary graph.

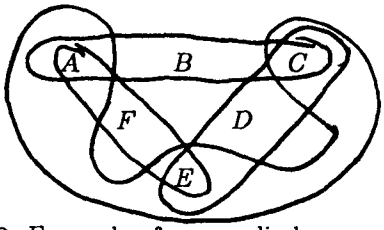

Fig. 2. Example of an acyclic hypergraph

A *block* of a re uced hypergraph is a connected, node-generated set of partial edges with no articulation set. A block is *trivial* if it consists of a single (partial) edge. A reduced hypergraph is *acyclic* if all its blocks are trivial; otherwise it is *cyclic*. A hypergraph is said to be cyclic or acyclic precisely if its reduction is.

Example 3: It is straightforward to verify that Fig. 2 shows an acyclic hypergraph. Its edges are ABC, CDE, EFA, and ACE. An articulation set for the set of all edges is $ABC \cap ACE = AC$, since the result of removing A and C from each edge is to leave the set of edges B, DE, EF, and E, which is not connected (B is disconnected from the others). Note that the set of edges $\{ABC, CDE, EFA\}$ has no articulation set. However, this set of edges is not node-generated, so there is no contradiction of our assertion that Fig. 2 is acyclic. []

We now define another equivalent notion of acylic hypergraphs, which we call "closed-acyclic." The advantage of dealing with the latter definition is that it is never necessary to consider partial edges that are not edges.

Let $(\mathcal{N}, \mathcal{E})$ be a hypergraph, and let $\mathcal{F}$ be a subset of $\mathcal{E}$. Let M be the set of nodes that is the union of the members of $\mathcal{F}$. We say $\mathcal{F}$ is *closed* if for each edge I of the hypergraph, there is an edge G in $\mathcal{F}$ such that $I \cap M \subseteq G$. Note that every closed set of edges is a node-generated set of partial edges, generated by M.

We say that a reduced hypergraph is *closed-acyclic* if every nontrivial, connected, closed set of edges has an articulation set. A hypergraph is said to be closed-acyclic precisely when its reduction is. Since every closed set of edges is a node-generated set of partial edges, it follows immediately that every acyclic hypergraph is closed-acyclic. The converse is also true and is proved in [BFMY].

Another equivalent notion of acyclicness in hypergraphs is captured by the following definition. The *graph of a hypergraph* H is the ordinary graph $G(H)$ with the same set of nodes as H and an edge between every two nodes that are in the same hyperedge of H.

We say that a hypergraph H is *chordal* if

1. $G(H)$ is chordal, and
2. For every set M of nodes that forms a clique in $G(H)$ there is a hyperedge of H that contains M.

III. Universal Relations

It is a convenience to the user if he can perceive the database as one relation over all the attributes. He is then spared concern over the details of database structure, specifically, which sets of attributes are actually used to head the columns of relations. The disadvantage is that unless additional semantic clues are available, the system cannot unambiguously interpret queries over the universal relation. [ABU] explains the rationale for the lossless join criterion for selecting interpretations of queries over the universal relation, [KU] explains how a language to query about the universal relation might be supported, and [Ke] and [BG1] give some of the problems that come up when trying to use the universal relation as a world view.

We claim that in almost any "real world" situation, a single join dependency suffices, together with some functional dependencies, to define the legal relations that might be the universal relation at some time. This assumption leads to a great simplification in the algorithms needed to interpret queries and to perform updates on the universal relation in a way that can be reflected in the actual relations of the database in a sensible manner. The following example shows how the single join dependency reflects much of our intuitive understanding of what the universal relation "means." Additional examples are given in [FMU]. Interestingly, the use of one join dependency to define the universal relation is very close to the ideas of [L1, L2] concerning what entity-relationship and Bachman diagrams mean as descriptions of the real world.

Example 4: Suppose we have attributes $CTHRSG$, standing for course, teacher, hour, room, student, and grade, respectively. The example comes from [U], where the intuition (not stated formally in these terms) is that there is a universal relation of 6-tuples

> $\{ cthrsg \mid t$ teaches c, c meets in room r at hour h, and student s is getting grade g in course $c \}$

It is shown in [FMU] that each such definition, where the universal relation is constrained to be defined as above for some values of the arbitrary predicates like "teaches," or "meets in$\cdots$at," is equivalent to requiring that the universal relation satisfy one join dependency, whose components are the sets of attributes over which the various predicates are defined. In this example, the universal relation must satisfy the join dependency $\bowtie(CT, CHR, CSG)$. This join dependency, plus a collection of functional dependencies, such as $C \rightarrow T$ and $CH \rightarrow R$, appear to characterize completely the structure of the universal relation in this example. It is our contention that this situation is the rule, rather than the exception. []

IV. The Main Theorem

Let us talk about a particular collection of sets of at-

tributes $R_1, \ldots, R_n$. Assume that the hypergraph has a set of nodes $\mathcal{N} = \bigcup_{i=1}^{n} R_i$ and set of edges $\mathcal{E} = \{ R_1, \ldots, R_n \}$. There are nine conditions that we shall show equivalent; some we have met, and some must be defined here.

Let $r_1, \ldots, r_n$ be an arbitrary list of relations over sets of attributes $R_1, \ldots, R_n$, respectively. We say that $r_1, \ldots, r_n$ are *pairwise consistent* if

$$\pi_{R_i \cap R_j}(r_i) = \pi_{R_i \cap R_j}(r_j)$$

for each i and j. We say that $r_1, \ldots, r_n$ are *globally consistent* if there is a relation r over

$$R = R_1 \cup \cdots \cup R_n$$

such that for each i, $r_i = \pi_{R_i}(r)$. That is, global consistency means that there is a universal relation of which all the r_i's are projections.

It is clear that global consistency implies pairwise consistency. However, the converse is false. Let r_1, r_2, and r_3 be the relations over sets of attributes AB, BC, and AC, respectively given by: $r_1 = \{ 00, 11 \}$, $r_2 = \{ 00, 11 \}$, and $r_3 = \{ 01, 10 \}$. It is easy to verify that these relations are pairwise consistent but not globally consistent.

We say *pairwise consistency is sufficient for global consistency* for a list $R_1, \ldots, R_n$ of sets of attributes, if every list $r_1, \ldots, r_n$ of relations over $R_1, \ldots, R_n$, respectively, is globally consistent if it is pairwise consistent. That is, pairwise and global consistency are equivalent for $R_1, \ldots, R_n$.

The *semijoin* of relations R and S is $\pi_R(R \bowtie S)$. A list of relations $(r_1, \ldots, r_n)$ over $(R_1, \ldots, R_n)$ is said to have a *full reducer* if there is a sequence of semijoins of the form $r_i := \pi_{R_i}(r_i \bowtie r_j)$ that converts all the r_i's into relations that are the projection of a single universal relation over $\mathcal{N}$.

A *join forest* for $(R_1, \ldots, R_n)$ is a forest with set of nodes $\{ R_1, \ldots, R_n \}$, such that

1. Each edge (R_i, R_j) is labeled by the set of attributes $R_i \cap R_j$, and
2. For all R_i and R_j, and for all A in $R_i \cap R_j$, there is some path between R_i and R_j such that each edge along the path has label A (possibly among others).

We say $(R_1, \ldots, R_n)$ has the *running intersection property* if we can order the R_i's as $S_1, \ldots, S_n$, such that for $2 \leq i \leq n$, there exists a $j_i < i$ such that

$$S_i \cap (S_1 \cup \cdots \cup S_{i-1}) \subseteq S_{j_i}$$

That is, the intersection of each S_i with the union of the previous S_j's is contained in one of these.

Theorem 1: The following conditions about $(R_1, \ldots, R_n)$ are equivalent.

1. $(\mathcal{N}, \mathcal{E})$ is an acyclic hypergraph.
2. $(\mathcal{N}, \mathcal{E})$ is a closed-acyclic hypergraph.
3. $(\mathcal{N}, \mathcal{E})$ is a chordal hypergraph.
4. The join dependency $\bowtie(R_1, \ldots, R_n)$ is equivalent to a set of multivalued dependencies.
5. $(R_1, \ldots, R_n)$ has the running intersection property.
6. The two operations of
   (i) delete from some R_i an attribute A that appears in no other R_j, and
   (ii) delete one R_i if there is an R_j, $i \neq j$, such that $R_i \subseteq R_j$,
   if applied repeatedly, reduce the list $(R_1, \ldots, R_n)$ to nothing.
7. Pairwise consistency is sufficient for global consistency for $R_1, \ldots, R_n$.
8. Every list of relations for $(R_1, \ldots, R_n)$ has a full reducer.
9. $(R_1, \ldots, R_n)$ has a join forest.

Proof: In the appendix, we give the details of the proof that (1) and (4) are equivalent. In ge eral, the various proofs that (1) implies the other conditions can be made by using the fact that any acyclic hypergraph can be broken into two or more pieces by an articulation set, and the resulting pieces, with the nodes of the articulation set restored, if necessary, to make full edges of the original hypergraph, are also acyclic. Thus, many properties follow by a "divide-and-conquer" type argument.

The equivalence of (1) and (4) is from [FMU]. [G] showed that (6) implies (7).† The equivalence of (8) and (9) is from [BG2]. The equivalence of (7) and (8) is from [H], while the remaining equivalences are from [BFMY]. []

We shall mention one other result, whose terms we shall not even define here, that relates the notion of acyclic hypergraphs to a concept that has appeared in the literature.

Theorem 2: A set of multivalued dependencies is "conflict free" in the sense of [L1] if and only if they are logically equivalent to a single join dependency with an acyclic hypergraph. []

V. Significance of Results

If the relations of a relational datatbase are defined over sets of attributes that have an acyclic hypergraph, then there is no ambiguity regarding interpretations of queries that connect two or more attributes. That is, there is a unique minimal set of relations that need to be joined to get a relation with a set of attributes that includes the attributes involved in the query. It is natural to suppose that this minimal set is the cor-

† It is interesting to observe that both [G] and, earlier [Z], were looking for a definition of "acyclic" that was equivalent to condition (7). They came up with more restictive versions of our notion of "acyclic hypergraph," principally because they considered not node-generated sets of edges, but rather only subsets of the given edges.

rect meaning of the query, and we therefore count this unambiguity as a beneficial property of sets of relation schemes with an acyclic hypergraph.

The desirability of sets of multivalued dependencies that are equivalent to a single join dependency was discussed in [Sc2]. The points made there depend on the detailed analysis of the way the semantic notion of "objects" interacts with multivalued dependencies and with the insertion and deletion of information in the database, which analysis was done in [Sc1]. The notion of "conflict freedom," from [L1], is another attempt to put restrictions on sets of multivalued dependencies that avoid problems in the definitions of how the database is to be updated, and also an attempt to to establish the equivalence between relational descriptions of the real world and descriptions in more "classical" terms, such as Bachman diagrams (directed graphs on collections of attributes).

Conditions (8) and (9) about full reducers and join trees were motivated not by issues of the structure of databases, but by the problem of implementing a query efficiently in a distributed database. On the assumption that each relation is at a different site, and communication must be minimized, it is frequently advantageous to perform semijoins first, sending only projections of relations from site to site, until the relations have been reduced to the point that every remaining tuple actually participates in the join with one or more tuples from the other relations. At that time, the reduced relations can be shipped to a single site, and their join taken [BG2].

Condition (7), that pairwise and global consistency be the same, was originally considered as a way of testing whether a collection of relations were the projection of a universal relation. The equivalence of (1) and (7) also says that if the relations of the database satisfy a join dependency with an acyclic hypergraph, we can maintain a universal relation, of which each database relation is the projection, if we agree that nulls will be used where necessary to fill out tuples of the universal relation, as described in numerous works on the subject, such as [KU, L3, M, Sc1, Vas, W]. When inserting or deleting from some relation R, we adjust the universal relation by considering interactions among the tuples of R and the other relations, inserting tuples with nulls into those relations only when necessary.† In the more general (not necessarily acyclic) case, the problem of adjusting the relations to maintain the property that they are the projection of a universal relation is NP-complete [HLY].

Finally, condition (6) is an efficient test for the acylicity property. Condition (5), the running intersection property, is a convenient tool for proving properties of acyclic hypergraphs, and the equivalence of (1), (2), (3), and (6) is an interesting graph-theoretic fact in its own right.

† Functional dependencies may cause us to discover that a null must really be the same as a value already existing in the database.

Appendix

The following is a proof that (1) is equivalent to (4) in Theorem 1.

Lemma 1: Let $j = \bowtie(R_1, \ldots, R_k)$ be a join dependency. Suppose that X and Y are disjoint sets of attributes. Then the multivalued dependency $X \twoheadrightarrow Y$ follows logically from j if and only if Y is the union of some connected components of the hypergraph of j with the set of nodes X deleted. []

Lemma 2: Let $H = (\mathcal{N}, \mathcal{E})$ be a hypergraph, $\mathcal{G}$ a closed subset of $\mathcal{E}$, and M the set of nodes of $\mathcal{G}$. Let $\mathcal{F}$ be a closed subset of the hypergraph $(M, \mathcal{G})$. Then $\mathcal{F}$ is also a closed subset of H. []

Let $\mathcal{F}$ be a closed connected set of two or more edges in an acyclic hypergraph. Since $\mathcal{F}$ is a node-generated set of partial edges (generated by the full set of nodes), there is an articulation set Q for $\mathcal{F}$. The edges in $\mathcal{E}$ can thus be partitioned into nonempty sets $\mathcal{H}_1, \ldots, \mathcal{H}_k$, $k \geq 2$, such that

1. each edge of $\mathcal{F}$ is in exactly one $\mathcal{H}_i$,
2. $\mathcal{H}_i - Q$† is connected for each i, and
3. if E_1 is an edge in $\mathcal{H}_i$ and E_2 is an edge in $\mathcal{H}_j$, $i \neq j$, then $E_1 \cap E_2 \subseteq Q$.

The fact that such a partition exists is exactly what we mean by saying that Q is an articulation set of $\mathcal{F}$. Each $\mathcal{H}_i$ is obtained by taking one of the connected components of $\mathcal{F} - Q$ and adding back to each edge its intersection with Q. We call each $\mathcal{H}_i$ a *component of $\mathcal{F}$ after articulation by Q*.

We now give the proof of the following.

Theorem 1: [(1) equivalent to (4)] MVD(j) is logically equivalent to j if and only if j's hypergraph is acyclic.

Proof: If: Evidently, j implies MVD(j), so we need only to prove that j follows from MVD(j). Without loss of generality, we assume that j's hypergraph is reduced. We use the tableau test of [MMS] and prove that if we chase the tableau of j with MVD(j), then we obtain a row of distinguished symbols ("a's") in all of the attributes.

We can assume that the hypergraph is connected. In proof, note that if the theorem holds for connected components, then, since by Lemma 1, $\emptyset \twoheadrightarrow X$ for every connected component X, we can form a row of all a's from the rows with a's in the connected components.

We shall now define, inductively on i, certain closed subsets of edges to be *i-level components* and *i-level augmented components*. We start the induction by letting the set of all edges be the unique 0-level component and the unique 0 level augmented component.

† We use $\mathcal{E} - Q$ to mean the set of partial edges generated from set of edges $\mathcal{E}$ by the set of nodes consisting of all the nodes in members of $\mathcal{E}$ except those in Q.

Suppose that we have defined $(i-1)$-level components and augmented components, $i \geq 1$. Assume inductively that each $(i-1)$-level augmented component is closed. We now define the i-level components and augmented components. If P is an $(i-1)$-level augmented component with exactly one edge, then P is also an i-level component and augmented component.

If P is an $(i-1)$-level augmented component with more than one edge, then since j is acyclic, there is an articulation pair (E, F) in P, since by the inductive hypothesis, P is closed, and therefore, node-generated. Let $\mathcal{H}_1, \ldots, \mathcal{H}_k$ be the components of P after articulation by $E \cap F$. For each m, $1 \leq m \leq k$, define $\mathcal{H}'_m$ to be $\mathcal{H}_m$ if $\mathcal{H}_m$ contains one or both of the edges E or F†, and $\mathcal{H}'_m = \mathcal{H}_m \cup \{E\}$ otherwise. Each $\mathcal{H}_m$ is an i-level component, and each $\mathcal{H}'_m$ is an i-level augmented component.

By construction, each $\mathcal{H}'_m$ contains either E or F. We now show that each $\mathcal{H}'_m$ is closed. It is sufficient, by Lemma 2, to show that $\mathcal{H}'_m$ is a closed subset of the closed subset P. Let M_m be the set of nodes appearing in $\mathcal{H}'_m$, and let I be an edge of P. We must show that $I \cap M_m \subseteq G$ for some G in $\mathcal{H}'_m$. If I is in $\mathcal{H}'_m$, then let $G = I$. Suppose I is not in $\mathcal{H}'_m$. We know that the unaugmented component $\mathcal{H}_m$ is a connected component of P after articulation by $E \cap F$. Thus, as I is in P but not in $\mathcal{H}'_m$, we know that its intersection with edges of $\mathcal{H}'_m$, if any, is limited to the articulation set, that is, $I \cap M \subseteq E \cap F$. Thus, we can take G to be whichever of E and F is in $\mathcal{H}'_m$.

We call each $\mathcal{H}'_m$ a *child* of P, and we call P the *father* of $\mathcal{H}'_m$. The transitive closure of the father relation is the *ancestor* relation. It is straightforward that each child has strictly fewer edges than its parent. It follows by induction on i that each i-level augmented component has at most $n - i$ edges, if n is the number of edges in the original hypergraph. In particular, the $(n-1)$-level augmented components are precisely the singleton sets $\{G\}$ for each edge G of the original hypergraph.

Assume as before that P is an $(i-1)$-level augmented component with (E, F) as articulation pair, and that $\mathcal{H}_1, \ldots, \mathcal{H}_k$, $k \geq 2$, are the components of P after articulation by $E \cap F$. We shall now show that

(*) (E, F) is an articulation pair for the whole hypergraph H, and that each of $\mathcal{H}_1, \ldots, \mathcal{H}_k$ is contained in a distinct component of H after articulation by $E \cap F$.

If $i = 1$, that is, if P is the 0-level augmented component, which contains every edge, then (*) is immediate. Assume that P has a father $\mathcal{R}$. We shall show that (E, F) is an articulation pair for $\mathcal{R}$, and that each of $\mathcal{H}_1, \ldots, \mathcal{H}_k$ is contained in a distinct component after articulation of $\mathcal{R}$ by $E \cap F$. Then (*) follows easily by induction on the level of P.

Assume by way of contradiction that $\mathcal{H}_1$ and $\mathcal{H}_2$ are in the same component after articulation of $\mathcal{R}$ by $Q = E \cap F$. Pick edge G_1 in H_1 and G_2 in H_2. We then know that there is a sequence of edges $X_1, \ldots, X_t$ of $\mathcal{R}$, where

1. $G_1 = X_1$,
2. $G_2 = X_t$, and
3. $X_m \cap X_{m+1}$ is not wholly contained within Q, for $1 \leq m < t$.

We know that some X_m is not in P, since H_1 and H_2 are distinct components of P after articulation by Q. Let u be the minimum value of m, and v the maximum value of m such that X_m is not in P. Then $1 < u \leq v < t$.

Let (C, D) be the articulation pair of $\mathcal{R}$ that led to the creation of P. Then P contains at least one of C and D; by renaming if necessary, assume it is C. Consider the sequence of edges

$$X_1, X_2, \ldots, X_{u-1}, C, X_{v+1}, \ldots, X_t$$

in which we have "spliced" C in place of $X_u, \ldots, X_v$. Every edge in this sequence is in P. To derive a contradiction, we need only show that this path is one in which each consecutive pair has a point in common that is not in $Q = E \cap F$, for we know that $X_1 = G_1$ and $X_t = G_2$, but G_1 and G_2 are supposed to be in distinct components of P after articulation by Q.

By condition (3), above, on the sequence of edges, we already know that

$$X_m \cap X_{m+1} - Q$$

is not empty for $1 \leq m < n - 1$ and for $v < m < t$. Thus, we need only show that sets $X_{u-1} \cap C - Q$ and $X_{v+1} \cap C - Q$ are nonempty. We know that there is some node d in $X_{u-1} \cap X_u - Q$, by condition (3) above. Now X_{u-1} and X_u are both in $\mathcal{R}$, but they are not in the same child of $\mathcal{R}$, since X_{u-1} is in P, but X_u is not. So, since d is in $X_{u-1} \cap X_u$, we know that d is in $C \cap D$. We already know that d is in $X_{u-1} \cap X_u - Q$, so d is in $X_{u-1} \cap C - Q$. Hence, $X_{u-1} \cap C - Q$ is nonempty. By an identical argument, $X_{v+1} \cap C - Q$ is nonempty, as was to be shown. We have now shown that if P is an $(i-1)$-level augmented component with (E, F) as articulation pair, and with $\mathcal{H}_1, \ldots, \mathcal{H}_k$ as the components of P after articulation by $E \cap F$, then (E, F) is an articulation pair for the whole hypergraph H, and that each of $\mathcal{H}_1, \ldots, \mathcal{H}_k$ is contained in a distinct component of H after articulation of H by $E \cap F$.

We now prove by reverse induction on i that if we chase the tableau of the join dependency j with

† Note that just because the removal of $E \cap F$ disconnects the hypergraph does not mean that E is disconnected from F. It is possible that $E \cap F$'s removal disconnects certain other edges whose sole connection to the rest of the graph was through nodes in $E \cap F$. It is true in this case, however, that another articulation pair could have been chosen.

MVD(j), then we obtain a row with distinguished symbols in all of the attributes of P, for each i-level augmented component P. The statement is true when $i = n - 1$, where n is the total number of nodes, since as we saw, every $(n-1)$-level augmented component is a single edge. We assume the result for i, $i \geq 1$, and show it for $i - 1$.

Let P be an $(i-1)$-level augmented component, and let $\mathcal{H}_1, \ldots, \mathcal{H}_k$ be its components after articulation as above, by $E \cap F$. Let $\mathcal{H}'_1, \ldots, \mathcal{H}'_k$ be the corresponding augmented components. By the inductive hypothesis, for each m, $1 \leq m \leq k$, there is a row r_m in the chased tableau with distinguished symbols in all the attributes of $\mathcal{H}'_m$. We just proved that $E \cap F$ is an articulation set of the whole hypergraph, and that each $\mathcal{H}_m$ is in a distinct component after articulation of the hypergraph by $E \cap F$. It follows easily that chasing the rows $r_1, \ldots, r_k$ using multivalued dependencies with left hand side $E \cap F$, we obtain a row with distinguished symbols in all the attributes of P.

This completes the induction step. In particular, it follows that there exists in the chased tableau a row with distinguished symbols in every attribute of the 0-level augmented component, that is, in every attribute whatsoever, as was to be shown.

Only if: Let the hypergraph of j be cyclic. Then there is a set M of nodes such that the set $R_1, \ldots, R_m$ of partial edges generated by M has no articulation set, and $m \geq 2$. We shall show by induction on the number of rows added when chasing the tableau of j according to the set of multivalued dependencies MVD(j), that the projection of any row onto the columns for M is the projection of a row that existed at the outset of the chase procedure.

The basis, zero rows, is immediate. For the induction, suppose that at some time we can use the multivalued dependency $X \twoheadrightarrow Y$ in MVD(j) to produce a row whose projection onto set of attributes M differs from the projections onto M that already exist. Let Z be all the attributes not in X or Y. Let r_1 and r_2 be the pair of rows to which $X \twoheadrightarrow Y$ is applied to generate, for the first time, a new projection onto M. So by the inductive assumption, $\pi_M(r_1) = \pi_M(s_1)$ and $\pi_M(r_2) = \pi_M(s_2)$ for some rows s_1 and s_2 of the original tableau. Now $\pi_M(r_1) \neq \pi_M(r_2)$, or else a new projection onto M could not be generated from these rows by a multivalued dependency. Thus $\pi_M(s_1) \neq \pi_M(s_2)$, and so $s_1 \neq s_2$.

Assume that s_1 corresponds to the set of attributes S_1 (that is, the distinguished symbols of s_1 appear in exactly the columns for the attributes in S_1), and similarly, s_2 corresponds to S_2. Then s_1 and s_2 agree precisely on $S_1 \cap S_2$. We know that $S_1 \cap M = R_p$ and that $S_2 \cap M = R_q$ for some p and q, because $R_1, \ldots, R_m$ is the node-generated set of partial edges generated by M. So $\pi_M(s_1)$ and $\pi_M(s_2)$ agree precisely on

$$S_1 \cap S_2 \cap M = R_p \cap R_q$$

Thus $\pi_M(r_1)$ and $\pi_M(r_2)$ agree precisely on $R_p \cap R_q$. To apply the multivalued dependency $X \twoheadrightarrow Y$ to r_1 and r_2 and get something new, rows r_1 and r_2 must at least agree on X. So $\pi_M(r_1)$ and $\pi_M(r_2)$ must agree on $M \cap X$. Since we showed that $\pi_M(r_1)$ and $\pi_M(r_2)$ agree precisely on $R_p \cap R_q$, it follows that $M \cap X \subseteq R_p \cap R_q$. Note that $R_p \neq R_q$, because if $R_p = R_q$ then $S_1 \cap M = S_2 \cap M$, whereupon $\pi_M(s_1) = \pi_M(s_2)$, a contradiction.

Since $R_1, \ldots, R_m$ forms a block, we know that deletion of $R_p \cap R_q$ does not disconnect these partial edges. But $M \cap X \subseteq R_p \cap R_q$. Thus, the deletion of X does not disconnect these partial edges. We may conclude from Lemma 1 that either $M \subseteq XY$ or $M \subseteq XZ$. As a result, applying the multivalued dependency $X \twoheadrightarrow Y$ produces only rows whose projection onto M already existed in another row, and it is not possible that this application of $X \twoheadrightarrow Y$ is the first to produce a row that has a's in a set of columns of M that is not a subset of the columns for any R_i.

We conclude that the chase process never introduces any new projections of rows onto M. Thus, each projected row has a's only in a set of columns that is contained in some R_i, $1 \leq i \leq m$. Unless M is a subset of an edge of the hypergraph for j, in which case the block is trivial, contrary to hypothesis, none of these rows have a's in all the rows of M. Thus certainly we do not, by chasing, produce a row with a's everywhere, and we conclude that j cannot be inferred from MVD(j). []

Acknowledgements

The authors would like to thank Maria Klawe and Moshe Vardi for their useful comments.

References

[ABU] Aho, A. V., C. Becri, and J. D. Ullman, "The theory of joins in relational databases," *ACM Transactions on Database Systems* 4:3 (1979), pp. 297–314.

[BBG] Beeri, C., P. A. Bernstein, and N. Goodman, "A sophisticate's introduction to database normalization theory," *Proc. International Conference on Very Large Data Bases*, pp. 113–124, 1978.

[BFH] Beeri, C., R. Fagin, and J. H. Howard, "A complete axiomatization for functional and multivalued dependencies," *ACM SIGMOD International Symposium on Management of Data*, pp. 47–61, 1977.

[BFMY] Beeri, C., R. Fagin, D. Maier, and M. Yannakakis, "On the desirable properties of acyclic database schemas," manuscript in preparation.

[BMSU] Beeri, C., A. O. Mendelzon, Y. Sagiv, and J. D. Ullman, "Equivalence of relational database schemes,"

Proc. Eleventh Annual ACM Symposium on the Theory of Computing, pp. 319–329, 1979.

[BV] Beeri, C. and M. Y. Vardi, "On the properties of join dependencies," *Proc. Workshop on Formal Bases for Databases*, Toulouse, Dec., 1979.

[Ber] Bernstein, P. A., "Synthesizing third normal form relations from functional dependencies," *ACM Transactions on Database Systems* 1:4 (1976), pp. 277–298.

[BG1] Bernstein, P. A. and N. Goodman, "What does Boyce-Codd normal form do?," *Proc. International Conference on Very Large Data Bases*, 1980.

[BG2] Bernstein, P. A and N. Goodman, "The theory of semijoins," TR CCA–79–27, Computer Corp. of America, Cambridge, Mass., 1979.

[C] Codd, E. F., "A relational model for large shared data banks," *Comm. ACM* 13:6 (1970), pp. 377–387.

[F1] Fagin, R., "Multivalued dependencies and a new normal form for relational databases," *ACM Transactions on Database Systems* 2:3 (1977), pp. 262–278.

[F2] Fagin, R., "Normal forms and relational database operators," *ACM SIGMOD International Symposium on Management of Data*, pp. 153–160, 1979.

[FMU] Fagin, R., A. O. Mendelzon, and J. D. Ullman, "A simplified universal relation assumption and its properties," RJ2900, IBM, San Jose, Calif., 1980.

[G] Graham, M. H., "On the universal relation," technical report, Univ. of Toronto, Sept., 1979.

[H] Honeyman, P., "Properties of the universal relation assumption," Ph. D. thesis, Princeton Univ., Princeton, N. J., 1980.

[HLY] Honeyman, P., R. E. Ladner, and M. Yannakakis, "Testing the universal instance assumption," *Inf. Proc. Letters*, 10:1 (1980), pp. 14–19.

[Ke] Kent, W., "Consequences of assuming a universal relation," IBM technical report, Dec., 1979, to appear in *TODS*.

[KU] Korth, H. F. and J. D. Ullman, "SYSTEM/U: a database system based on the universal relation assumption," *Proc. XP1 Conference*, Stonybrook, N. Y., June, 1980.

[L1] Lien, Y. E., "On the equivalence of database models," private communication, June, 1980.

[L2] Lien, Y. E., "On the semantics of the entity-relationship model," in *Entity-Relationship Approach to Systems Analysis and Design* (P. P. Chen, ed.), pp. 155-167, North Holland, Amsterdam, 1980.

[L3] Lien, Y. E., "Multivalued dependencies with null values in relational data bases," *Proc. International Conference on Very Large Data Bases*, pp. 61–66, 1979.

[M] Maier, D., "Discarding the universal instance assumption: preliminary results," *Proc. XP1 Conference*, Stonybrook, N. Y., June, 1980.

[MMS] Maier, D., Y. Sagiv, and A. O. Mendelzon, "Testing implications of data dependencies," *ACM Transactions on Database Systems* 4:4 (1979), pp. 455–469.

[R1] Rissanen, J., "Theory of joins for relational databases—a tutorial survey," *Proc. Seventh Symp. on Mathematical Foundations of Computer Science*, Lecture Notes in CS, 64, Springer-Verlag, pp. 537–551.

[R2] Rissanen, J., "Independent components of relations," *ACM Transactions on Database Systems* 2:4 (1977), pp. 317–325.

[Sc1] Sciore, E., "Null values, updates, and normalization in relational databases," doctoral dissertation, Princeton Univ., Princeton, N. J., 1980.

[Sc2] Sciore, E., "Some observations on real-world data dependencies," *Proc. XP1 Conference*, Stonybrook, N. Y., June, 1980.

[U] Ullman, J. D., *Principles of Database Systems*, Computer Science Press, Potomac, Md., 1980.

[Var] Vardi, M. Y., "Inferring multivalued dependencies from functional and join dependencies," Dept. of Applied Math., Weizmann Inst. of Science, Rehovot, Israel, 1980.

[Vas] Vassilou, Y., "Null values in database management—a denotational semantics approach," *ACM SIGMOD International Symposium on Management of Data*, pp. 162-169, 1979.

[W] Walker, A., "Time and space in a lattice of universal relations with blank entries," *Proc. XP1 Conference*, Stonybrook, N. Y., June, 1980.

[Z] Zaniolo, C., "Analysis and design of relational schemata for database systems," Ph. D. thesis, UCLA, 1976.