On the Desirability of Acyclic Database Schemes

CATRIEL BEERI

The Hebrew University of Jerusalem, Jerusalem, Israel

RONALD FAGIN

IBM Research Laboratory, San Jose, California

DAVID MAIER

State University of New York at Stony Brook, Stony Brook, New York

AND

MIHALIS YANNAKAKIS

Bell Laboratories, Murray Hill, New Jersey

Abstract. A class of database schemes, called acyclic, was recently introduced. It is shown that this class has a number of desirable properties. In particular, several desirable properties that have been studied by other researchers in very different terms are all shown to be equivalent to acyclicity. In addition, several equivalent characterizations of the class in terms of graphs and hypergraphs are given, and a simple algorithm for determining acyclicity is presented. Also given are several equivalent characterizations of those sets M of multivalued dependencies such that M is the set of multivalued dependencies that are the consequences of a given join dependency. Several characterizations for a conflict-free (in the sense of Lien) set of multivalued dependencies are provided.

Categories and Subject Descriptors. F.4.1 **[Mathematical Logic and Formal Languages]**: Mathematical Logic; G.2.2 **[Discrete Mathematics]**. Graph Theory—*graph algorithms; trees*, H.2.1 **[Database Management]**: Logical Design—*normal forms; schema and subschema*; H.3.3 **[Information Storage and Retrieval]**: Information Search and Retrieval—*query formulation*

General Terms Algorithms, Design, Languages, Management, Theory

Additional Key Words and Phrases: Acyclicity, hypergraph, database scheme, relational database, multivalued dependency, join dependency, conflict-freedom

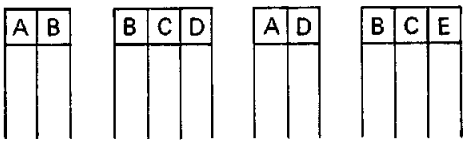
1. *Introduction*

In the relational model of data, as defined by Codd [14], an arbitrary database scheme is possible. A *database scheme* can be thought of as a collection of table

The work of the first author was performed at IBM Research Laboratory, San Jose, and at Stanford University and was supported in part by the National Science Foundation under Grant MCS 79-04528 The work of the third author was supported in part by the National Science Foundation under Grant IST 79-18264.

Authors' addresses: C. Beeri, The Hebrew University of Jerusalem, Jerusalem, Israel; R. Fagin, IBM Research Laboratory K51/281, 5600 Cottle Road, San Jose, CA 95193, D Maier, Oregon Graduate Center, 19600 NW Walker Road, Beaverton, OR 97006; M Yannakakis, Bell Laboratories, 600 Mountain Avenue, Murray Hill, NJ 07974

FIGURE 1



skeletons (as in Figure 1), or, alternatively, as a set of subsets of attributes. For example, if $\mathcal{N} = \{A, B, C, D\}$ (or, for short, simply $ABCD$) is the set of attributes, or column names, then one example of a database scheme is $\{AB, BCD, AD, BCE\}$. This database scheme corresponds to having four relations in the database, as in Figure 1. The first relation has columns A and B, the second has columns B, C, D, and so on.

Beeri et al. [6] introduced a special class of database schemes, called *acyclic*. Fagin et al. [17] have shown that this class enjoys a certain desirable property (which we describe later). Our goal in this paper is to identify a number of other desirable properties, which have been studied by other researchers in quite different contexts, and show that each of these properties is equivalent to acyclicity. Thus the class of acyclic database schemes is a natural, important class, since it can be characterized in a number of ways, each corresponding to a desirable property of database schemes or to a natural graph-theoretic property. As we shall see, there are various undesirable and pathological phenomena that can take place for general database schemes but not for acyclic database schemes. So, by restricting our attention to the acyclic case, the theory is more elegant. Furthermore, it has been conjectured [17] that acyclic database schemes are sufficiently general to encompass most "real-world" situations. At the very least, database designers should be aware of acyclicity and strive for it. Assuming the conjecture, it follows that by focusing on the acyclic case, researchers can develop a powerful, elegant theory that often applies to "real-world" schemes. For acyclic schemes there are efficient (polynomial-time) algorithms for solving problems that are NP-complete in the unrestricted case. We shall give one such example (determining global consistency); other examples are shown by Yannakakis [43]. Further, we shall give a simple algorithm for determining acyclicity.

There are various interesting problems concerning relational databases where some type of object can be viewed as a collection of sets, and a property of the object depends on the structure of this collection. Now a collection of sets can be viewed as being a hypergraph. It turns out that for various properties $\mathcal{P}$, acyclicity of the hypergraph is equivalent to $\mathcal{P}$ holding. Such properties $\mathcal{P}$ occur in (at least) three distinct areas. The first area arises when a database scheme is viewed as a collection of attribute sets. We shall discuss certain properties of relational databases that depend on the structure of the scheme (one such property is whether or not every pairwise consistent database over the scheme is globally consistent). A second area is the theory of dependencies. One of the important types of dependencies is the join dependency, which can be viewed as a collection of sets. The desired property here is that the join dependency is logically equivalent to a collection of binary join dependencies (i.e., multivalued dependencies). A third area is query processing. Here, join expressions are of importance, and these again are collections of sets. The interesting problems are the existence of time-efficient and/or space-efficient access paths. All these problems from distinct areas are linked together by acyclicity conditions on a hypergraph structure.

In Section 2 we present definitions. In Section 3 we define a number of conditions that are equivalent to acyclicity. In Section 4 we discuss the significance of our results and their relationship to other work. In Section 5 we discuss several other types of

acyclicity for hypergraphs. Our main theorem, that the various properties discussed in Section 3 are all equivalent to acyclicity, is proved in Section 6. In Section 7 we give several characterizations of those sets M of multivalued dependencies such that M is the set of multivalued dependencies that are the consequences of a given join dependency. In Section 8 we give several characterizations for a conflict-free (in the sense of [26]) set of multivalued dependencies. We also show that an arbitrary acyclic join dependency $\bowtie\{R_1, \ldots, R_n\}$ is equivalent to a conflict-free set of at most $n - 1$ multivalued dependencies. This strengthens the result of [17] that each acyclic join dependency is equivalent to a set of multivalued dependencies whose size is polynomial in the size of the join dependency.

2. *Definitions*

Let $\mathcal{N}$ be a finite set of distinct symbols, called *attributes* (or *column names*), and let Y be a subset of $\mathcal{N}$. In the spirit of Armstrong [2] and of Aho et al. [1] we define a *Y-tuple* (or simply a *tuple*, if Y is understood) to be a function with domain Y. Thus a tuple is a mapping that associates a value with each attribute in Y. If X is a subset of Y and t is a Y-tuple, then $t[X]$ denotes the X-tuple obtained by restricting the mapping to X. A *Y-relation* (or a *relation over Y*, or simply a *relation*, if Y is understood), is a finite set of Y-tuples. If r is a Y-relation and X is a subset of Y, then by $r[X]$, the *projection* of r onto X, we mean the set of all tuples $t[X]$, where t is in r. We shall usually denote sets of attributes by uppercase letters and relations by lowercase letters.

If $\mathcal{N}$ is a set of attributes, then we define a *database scheme* $\mathbf{R} = \{R_1, \ldots, R_n\}$ to be a set of subsets of $\mathcal{N}$. Intuitively, for each i, the set R_i of attributes is considered the set of column names for a relation. We may call the R_i's *relation schemes*. If $r_1, \ldots, r_n$ are relations, where r_i is a relation over R_i $(1 \leq i \leq n)$, then we call $\mathbf{r} = \{r_1, \ldots, r_n\}$ a *database* over $\mathbf{R}$.

If r is a relation over R, and X and Y are subsets of R, then we say [15] that the *multivalued dependency* (*MVD*) $X \twoheadrightarrow Y$ holds for r if whenever t_1 and t_2 are tuples of r with $t_1[X] = t_2[X]$, then there exists a tuple t_3 of r such that

(1) $t_3[X] = t_1[X] = t_2[X]$,
(2) $t_3[Y] = t_1[Y]$, and
(3) $t_3[R - XY] = t_2[R - XY]$.

Intuitively, the set of Y-values associated with each given X-value is independent of the values in all other attributes. By XY in (3) above, we mean $X \cup Y$.

Let $\mathbf{r} = \{r_1, \ldots, r_n\}$ be a database over $\mathbf{R}$. The *join* of the relations $\mathbf{r}$ (where the join is denoted by either $r_1 \bowtie \cdots \bowtie r_n$ or $\bowtie\mathbf{r}$) is the set of all tuples t with attributes $R_1 \cup \cdots \cup R_n$, such that $t[R_i]$ is in r_i for each i.

We say [1, 32] that a relation r with attributes $R_1 \cup \cdots \cup R_n$ obeys the *join dependency* $\bowtie\{R_1, \ldots, R_n\}$ if $r = \bowtie\{r_1, \ldots, r_n\}$, where $r_i = r[R_i]$, for $1 \leq i \leq n$. It follows that the join dependency $\bowtie\{R_1, \ldots, R_n\}$ holds for the relation r if and only if r contains each tuple t for which there are tuples $w_1, \ldots, w_n$ of r (not necessarily distinct) such that $w_i[R_i] = t[R_i]$ for each i $(1 \leq i \leq n)$. As an example, the relation r in Figure 2 violates the join dependency $\bowtie\{AB, ACD, BC\}$. For, let w_1, w_2, w_3 be, respectively, the tuples (0, 1, 0, 0), (0, 2, 3, 4), and (5, 1, 3, 0) of r; let R_1, R_2, R_3 be, respectively, AB, ACD, and BC; and let t be the tuple (0, 1, 3, 4); then $w_i[R_i] = t[R_i]$ for each i $(1 \leq i \leq n)$, although t is not a tuple in the relation r. However, it is straightforward to verify that the same relation r obeys, for example, the join dependency $\bowtie\{ABC, BCD, ABD\}$.

FIGURE 2

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

Let Σ be a set of dependencies, and let σ be a single dependency. When we say that Σ *logically implies* σ or that σ *is a logical consequence of* Σ, we mean that whenever every dependency in Σ holds for a relation r, then σ also holds for r. That is, there is no "counterexample relation" or "witness" r such that every sentence in Σ holds for r but σ does not hold for r. We write $\Sigma \models \sigma$ to mean that Σ logically implies σ. For example, $\{A \twoheadrightarrow B, B \twoheadrightarrow C\} \models A \twoheadrightarrow C$.

3. *Conditions Equivalent to Acyclicity*

Let $\mathbf{R} = \{R_1, \ldots, R_n\}$ be a database scheme, as defined in the introduction. Thus there is a fixed set $\mathcal{N}$ of attributes, and $R_i \subseteq \mathcal{N}$ for each i. We always assume that every attribute of $\mathcal{N}$ appears in at least one R_i. We now consider a number of conditions on $\mathbf{R}$, all of which will turn out to be equivalent.

Condition 3.1. $\mathbf{R}$ is an acyclic hypergraph.

A *hypergraph* is a pair $(\mathcal{N}, \mathcal{E})$, where $\mathcal{N}$ is a finite set of nodes and $\mathcal{E}$ is a set of edges (or hyperedges) which are arbitrary subsets of $\mathcal{N}$. An ordinary undirected graph (without self-loops) is, of course, a hypergraph whose every edge is of size two.

The hypergraph of a database scheme $\{R_1, \ldots, R_n\}$ has as its set of nodes those attributes that appear in one or more of the R_i's, and as its set of edges $\mathbf{R} = \{R_1, \ldots, R_n\}$. We shall often speak of the "hypergraph $\mathbf{R}$" without mentioning the set $\mathcal{N}$ of nodes, since, as noted, we tacitly assume that $\mathcal{N} = \bigcup\{R_i : 1 \leq i \leq n\}$.

Let us give some terminology for hypergraphs. A *path* from node s to node t is a sequence of $k \geq 1$ edges $E_1, \ldots, E_k$ such that

(1) s is in E_1,
(2) t is in E_k, and
(3) $E_i \cap E_{i+1}$ is nonempty if $1 \leq i < k$.

We also say the above sequence of edges is an *edge path* (or just *path* when no confusion arises) from E_1 to 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. The *connected components* are the maximal connected sets of edges.

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. If we say that $\bowtie\{R_1, \ldots, R_n\}$ (or, for short, $\bowtie\mathbf{R}$) is the join dependency *corresponding* to the hypergraph $\{R_1, \ldots, R_n\}$, then the join dependency corresponding to a hypergraph and the join dependency corresponding to its reduction are logically equivalent [7].

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 defined to be obtained by intersecting the edges in $\mathcal{E}$ with $\mathcal{M}$, that is, taking $\{E \cap \mathcal{M} : E \in \mathcal{E}\} - \{\emptyset\}$ and then taking the reduction of this set. 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.*

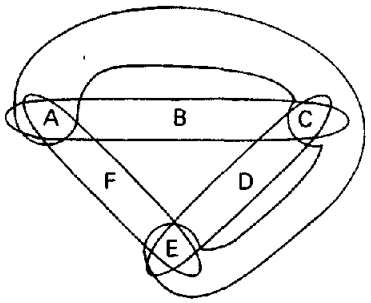


FIGURE 3

Let $\mathcal{F}$ be a connected, reduced 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 of $\mathcal{F}$, that is, $\{D - Q : D \in \mathcal{F}\} - \{\varnothing\}$, is not a connected set of partial edges. More generally, if $\mathcal{F}$ is a (not necessarily connected) set of partial edges and E and F are in $\mathcal{F}$, then we say that (E, F) is an articulation pair, and that $Q = E \cap F$ is an articulation set of $\mathcal{F}$, if the result of removing Q from every edge in $\mathcal{F}$ strictly increases the number of connected components of $\mathcal{F}$. It is clear that an articulation set in a hypergraph is a generalization of the concept of an articulation point in an ordinary graph.

A *block* of a reduced hypergraph is a connected, node-generated set of partial edges with no articulation set. A set is *trivial* if it contains less than two members. 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 3.1. It is straightforward to verify that Figure 3 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 is not node generated, so there is no contradiction of our assertion that the hypergraph of Figure 3 is acyclic. □

Condition 3.2. **R** is a closed-acyclic hypergraph.

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

Recall that a reduced hypergraph is acyclic if every nontrivial, connected, node-generated set of partial edges has an articulation set. 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 if 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. We shall show that, in fact, "acyclic" and "closed-acyclic" are equivalent. The intuitive advantage of dealing with the closed-acyclic definition rather than with the acyclic definition is that under the closed-acyclic definition it is not necessary to consider partial edges that are not edges.

Condition 3.3. **R** is a chordal, conformal hypergraph.

We begin with some definitions for (ordinary, undirected) graphs. A *clique* in a graph is a set of nodes such that every pair forms an edge of the graph. A *cycle* in a

graph is a sequence $(a_1, \ldots, a_m)$ of nodes, $m \geq 3$, such that

(i) each a_i is distinct, except that $a_1 = a_m$, and
(ii) (a_i, a_{i+1}) is an edge for $1 \leq i < m$.

Let $\mathcal{H}$ be a hypergraph. The *graph* $G(\mathcal{H})$ of $\mathcal{H}$ has the same nodes as $\mathcal{H}$ and an edge between every pair of nodes that are in the same hyperedge of $\mathcal{H}$. Thus, the edges of $G(\mathcal{H})$ are precisely the set of all pairs (a, b) for which there is a hyperedge E of $\mathcal{H}$ that contains both a and b (and possibly other nodes).

A hypergraph $\mathcal{H}$ is *conformal* [10] if for every clique V in $G(\mathcal{H})$ there is a hyperedge of $\mathcal{H}$ that contains V. We now prove the following simple characterization of reduced, conformal hypergraphs.

THEOREM 3.2. *Hypergraph $\mathcal{H}$ is reduced and conformal if and only if its hyperedges are precisely the maximal cliques of a graph. If there is such a graph, then the graph is $G(\mathcal{H})$.*

PROOF. ($\Rightarrow$): Let $\mathcal{H}$ be a reduced, conformal hypergraph, and let $G = G(\mathcal{H})$ be its graph. We now show that the hyperedges of $\mathcal{H}$ are precisely the maximal cliques of G. Let V be a hyperedge of $\mathcal{H}$. By definition of G it follows that V is a clique of G. If V were not a maximal clique of G, then there would be a clique W of G that properly contains V. Since $\mathcal{H}$ is a conformal hypergraph, there is a hyperedge X of $\mathcal{H}$ that contains W. But then hyperedge X properly contains hyperedge V; this contradicts our assumption that $\mathcal{H}$ is reduced.

($\Leftarrow$): Let $\mathcal{H}$ be a hypergraph whose hyperedges are precisely the maximal cliques of a graph D. It is clear that $\mathcal{H}$ is reduced; we shall show that it is conformal. Further, let $G = G(\mathcal{H})$ be the graph of $\mathcal{H}$; we shall show that $G = D$.

Now $(a, b) \in D$ if and only if $\{a, b\}$ is in a maximal clique of D, if and only if $\{a, b\}$ is a subset of an edge of $\mathcal{H}$, if and only if $(a, b) \in G$. Hence, $G = D$. Since $G = D$, it follows by assumption that the hyperedges of $\mathcal{H}$ are precisely the maximal cliques of $G = G(\mathcal{H})$. In particular, for every clique V in $G(\mathcal{H})$ there is a hyperedge of $\mathcal{H}$ that contains V. Thus $\mathcal{H}$ is conformal. □

A graph is *chordal* [18] if every cycle with at least four distinct nodes has a chord, that is, an edge connecting two nonconsecutive nodes of the cycle. Chordal graphs are sometimes called *triangulated*. A hypergraph $\mathcal{H}$ is *chordal* if its graph $G(\mathcal{H})$ is chordal. We note that in [6] a hypergraph is called chordal if it is not only chordal (under our definition), but also conformal. We have decided that it is useful to change this convention. We note some important, well-known properties of chordal graphs.

(1) Chordality of graphs is a *hereditary* property; that is, deleting nodes (and their incident edges) from a chordal graph leaves a subgraph that is also chordal.
(2) Every chordal graph has a node v whose neighborhood is a clique; that is, there is an edge between every pair of nodes, each of which is adjacent to v. Such a node is called *simplicial* (e.g., see [18]).

Condition 3.4. Graham's algorithm succeeds with input **R**.

Graham's algorithm [21] applies the following two operations to $\mathbf{R} = \{R_1, \ldots, R_n\}$ repeatedly until neither can be applied:

(a) If A is an attribute that appears in exactly one R_i, then delete A from R_i.
(b) Delete one R_i if there is an R_j with $j \neq i$ such that $R_i \subseteq R_j$.

Intuitively, operations of type (a) remove attributes that cannot have any effect on cyclicity or acyclicity, and operations of type (b) cause a hypergraph to be replaced by its reduction.

The algorithm *succeeds* if it terminates with the empty set; otherwise, it *fails*. We note that it is not hard to show that the algorithm is Church–Rosser. That is, the set that the algorithm terminates with is independent of the sequence of steps taken in executing the algorithm but depends only on the input.

Example 3.3. Let us apply Graham's algorithm to the hypergraph of Example 3.1, with edges ABC, CDE, EFA, and ACE. Nodes B, D, and F each appear in only one edge, and so they are each deleted by applications of rule (a) of Graham's algorithm. We are then left with edges AC, CE, EA, and ACE. Now edge AC is a subset of edge ACE, so by an application of rule (b) of the algorithm, this edge is deleted. This leaves us with edges CE, EA, and ACE. Similarly, edges CE and EA are deleted by applications of rule (b). We are then left with only one edge, namely ACE. Each of the nodes A, C, and E now appears in only one edge, and so by applications of rule (a), each of them is deleted. We are left with the empty set, and so Graham's algorithm succeeds. □

Condition 3.5. The join dependency $\bowtie\mathbf{R}$ is equivalent to a set of multivalued dependencies.

Since multivalued dependencies are simpler than join dependencies (they are special cases of join dependencies), it is a desirable property of a join dependency for it to be equivalent to a set of multivalued dependencies. Moreover, it is easy to test (by sorting and counting) whether a given multivalued dependency holds for a relation; however, the problem of whether a given join dependency holds for a given relation is NP-complete [24, 30].

Fagin et al. [17] showed that if a join dependency is equivalent to a set of multivalued dependencies, then it is equivalent to a set M of multivalued dependencies whose size is polynomial in the size of the join dependency. In Section 8 we strengthen this result by showing that if the join dependency is $\bowtie\{R_1, \ldots, R_n\}$, then the set M can be taken to be a set of at most $n - 1$ multivalued dependencies.

Condition 3.6. The join dependency $\bowtie\mathbf{R}$ is equivalent to a conflict-free set of multivalued dependencies.

We define *conflict free* in Section 8.

Condition 3.7. Every pairwise consistent database over $\mathbf{R}$ is globally consistent.

Let r and s be relations with attributes R and S, respectively, and let $Q = R \cap S$. Thus Q is precisely the set of attributes that r and s have in common. We say that r and s are *consistent* if $r[Q] = s[Q]$, that is, the projections of r and s onto their common attributes are the same.

Let $\mathbf{r} = \{r_1, \ldots, r_n\}$ be an arbitrary database over $\mathbf{R} = \{R_1, \ldots, R_n\}$. We say that $\mathbf{r}$ is *pairwise consistent* if each pair r_i and r_j is consistent, that is, if $r_i[R_i \cap R_j] = r_j[R_i \cap R_j]$ for each i and j. We say that $\mathbf{r}$ is *globally consistent* if there is a relation r over attributes $\mathcal{N} = R_1 \cup \cdots \cup R_n$ such that $r_i = r[R_i]$ for each i. Thus $\mathbf{r}$ is globally consistent if there is a "universal relation" r such that each r_i is a projection of r. It is known [1] that if there is such a universal relation r, then $\bowtie\mathbf{r}$ is also such a universal relation. Note that a pair of relations is globally consistent if and only if the two relations are consistent. Rissanen [33] calls a globally consistent set of relations

FIGURE 4

A	B
0	0
1	1

B	C
0	0
1	1

A	C
0	1
1	0

joinable. Joinability is a critical assumption in Rissanen's theory of independent components of relations. A globally consistent database is also called *join compatible* [8], *join consistent* [24], *valid* [34], *consistent* [16], or *decomposed* [40].

It is clear that if **r** is globally consistent, then it is pairwise consistent. If $n = 2$, that is, if only two relations are involved, then, as noted above, the converse is true. However, in general, the converse is false. For example, let r_1, r_2, and r_3 be the three relations in Figure 4, over attributes AB, BC, and AC, respectively. It is easy to verify that these relations are pairwise consistent but not globally consistent.

We say that *every pairwise consistent database over* **R** *is globally consistent* if for every database **r** over **R**, pairwise consistency of **r** implies global consistency of **r** (and thus pairwise consistency and global consistency are equivalent for **r**).

Honeyman et al. [24] have shown that the problem of deciding whether a database **r** is globally consistent is NP-complete. However, if every pairwise consistent database over **R** is globally consistent, then there is a simple polynomial-time test for global consistency, namely, pairwise consistency.

By our definitions, a database is required to contain a finite number of tuples. We could define a *possibly infinite database* by removing this restriction. It is then not obvious that the following two statements are equivalent:

(a) Every pairwise consistent database over **R** is globally consistent.
(b) Every pairwise consistent, *possibly infinite* database over **R** is globally consistent.

However, it follows from our proof of Theorem 3.4 below that (a) and (b) are indeed equivalent.

Condition 3.8. Every database over **R** has a full reducer.

The *semijoin* [11, 12] $r \ltimes s$ of relations r and s (over attributes R and S, respectively) is $(r \bowtie s)[R]$. A *semijoin program* is a sequence of semijoin statements $r_i := r_i \ltimes r_j$. A *full reducer* for a database **r** is a semijoin program that converts **r** into a globally consistent database.

If it is necessary to join a number of relations, each of which is at a different site, and if the amount of communication is to be minimized, then it is frequently advantangeous to perform semijoins first, by sending only certain projections of relations from site to site, until the relations have been pruned (by removing tuples) 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 pruned relations can be shipped to a single site and their join taken [12].

Condition 3.9. **R** has a join tree.

A *join tree* for **R** is a tree with set **R** of nodes, such that

(1) Each edge (R_i, R_j) is labeled by the set of attributes $R_i \cap R_j$, and
(2) For every pair R_i, R_j $(R_i \neq R_j)$ for every A in $R_i \cap R_j$, each edge along the unique path between R_i and R_j includes label A (possibly among others). We call this path an *A-labeled path*.

A join tree is so named because it yields a "good" (in a manner to be defined soon) way to join together all of the relations. For, let T be a join tree for **R** =

$\{R_1, \ldots, R_n\}$. Select a root for the tree T. Let $S_1, \ldots, S_n$ be $R_1, \ldots, R_n$ ordered by increasing depth. Thus, if S_j is the parent of S_i, then $j < i$. A "good" way to join together all of the relations is first to take the join of the S_1 and S_2 relations, then join the result with the S_3 relation, then join the result with the S_4 relation, and so on. "Good" means that if the database is pairwise consistent, then by joining the relations in this manner, the number of tuples grows monotonically (this fact follows from our proof of Theorem 3.4 below; see Condition 3.11 below for an explanation of monotonicity).

Condition 3.10. **R** has the running intersection property.

We say that **R** has the *running intersection property* if there is an ordering $R_1, \ldots, R_n$ of **R** such that for $2 \leq i \leq n$ there exists $j_i < i$ such that $R_i \cap (R_1 \cup \cdots \cup R_{i-1}) \subseteq R_{j_i}$. That is, the intersection of each R_i with the union of the previous R_j's is contained in one of these.

Condition 3.11. **R** has a monotone join expression.

Consider the following scenario. A user desires to take the join of four relations r_1, r_2, r_3, and r_4. The following might happen. He might first form $r_1 \bowtie r_2$, which might have, say, a thousand tuples. Then he might join the result with r_3, to obtain $r_1 \bowtie r_2 \bowtie r_3$, a relation with, say, a million tuples. He might finally join the result with r_4, to obtain his desired answer $r_1 \bowtie r_2 \bowtie r_3 \bowtie r_4$, which might have only ten tuples. Thus, even though the result he was seeking had only ten tuples, he might have had an intermediate result with a million tuples. We now discuss "monotone join expressions," which prevent this unpleasant behavior.

A *join expression* is a well-formed expression formed out of relation schemes, the symbol "$\bowtie$," and parentheses, in which every join is binary. For example, if R_1, R_2, R_3, and R_4 are among the relation schemes, then $((R_2 \bowtie R_3) \bowtie (R_1 \bowtie R_4))$ is a join expression, which corresponds to joining the R_2 and the R_3 relations, joining the R_1 and R_4 relations, and then joining together the two results.

Let θ be a join expression whose relation schemes are all in **R**, and let **r** be a database over **R**. By $\theta(\mathbf{r})$ we mean the relation that results by replacing each relation scheme R in θ by r, where $r \in \mathbf{r}$ and r has attributes R. For example, if $\mathbf{r} = \{r_1, r_2, r_3, r_4\}$ and θ is the join expression $(R_2 \bowtie (R_3 \bowtie R_2))$, where r_2 and r_3 have attributes R_2 and R_3, respectively, then $\theta(\mathbf{r})$ is the relation $(r_2 \bowtie (r_3 \bowtie r_2))$, that is, the relation $r_2 \bowtie r_3$.

A *subexpression* of a join expression is defined in the usual way. Let θ be a join expression containing relation schemes **R**, and let **r** be a database over **R**. We say that θ is *monotone with respect to* **r** if for every subexpression $(\theta_1 \bowtie \theta_2)$ of θ, the relations $\theta_1(\mathbf{r})$ and $\theta_2(\mathbf{r})$ are consistent. Intuitively, θ is monotone with respect to **r** if no tuples are lost in taking any of the binary joins obtained by "executing" $\theta(\mathbf{r})$ as dictated by the parentheses. (We say that *no tuples are lost* in taking the join of relations r and s if r and s are each projections of $r \bowtie s$, i.e., if r and s are consistent.) As an example, $((R_2 \bowtie R_3) \bowtie (R_1 \bowtie R_4))$ is monotone with respect to $\mathbf{r} = \{r_1, r_2, r_3, r_4\}$, where r_i has attributes R_i $(1 \leq i \leq 4)$, if

(a) r_2 and r_3 are consistent,
(b) r_1 and r_4 are consistent, and
(c) $(r_2 \bowtie r_3)$ and $(r_1 \bowtie r_4)$ are consistent.

We say that θ is *monotone* if it is monotone with respect to every pairwise consistent database over **R**. If θ involves precisely the relation schemes **R**, then we say that **R** *has a monotone join expression.* Monotone join expressions provide a "space-efficient"

manner for taking a join, in that no "intermediate" join has more tuples than the "final" join $r_1 \bowtie \cdots \bowtie r_n$.

Condition 3.12. **R** has a monotone, sequential join expression.

Certain join expressions, called sequential join expressions, are of special interest. Let θ be a join expression over **R**. If θ is of the form $(\cdots ((R_1 \bowtie R_2) \bowtie R_3) \cdots \bowtie R_n)$, where $R_1, \ldots, R_n$ is an ordering of the distinct members of **R**, then we say that θ is *sequential*. Intuitively, a sequential join expression $(\cdots ((R_1 \bowtie R_2) \bowtie R_3) \cdots \bowtie R_n)$ corresponds to first joining the R_1 and the R_2 relations, then joining the result with the R_3 relation, then joining the result with the R_4 relation, and so on.

Saying that there is a monotone, sequential join expression over **R** means that there is an ordering $R_1, \ldots, R_n$ of **R** such that if $\mathbf{r} = \{r_1, \ldots, r_n\}$ is a pairwise consistent database over **R**, then the join $r_1 \bowtie \cdots \bowtie r_i$ is consistent with r_{i+1} $(1 \leq i < n)$. Thus, if we first join r_1 with r_2, join the result with r_3, join the result with r_4, and so on, then no tuples are lost in taking any of the joins; hence the number of tuples grows monotonically. As we noted, having a monotone join expression (Condition 3.11 above) guarantees that no intermediate join has more tuples than the final join. Having a monotone, *sequential* join expression has the further advantage that only one intermediate join needs to be maintained at a time.

The main result of this paper is that each of the above conditions on **R** are equivalent. Thus we shall prove the following theorem.

THEOREM 3.4. *The following conditions on* **R** *are equivalent*:

(1) **R** *is an acyclic hypergraph.*
(2) **R** *is a closed-acyclic hypergraph.*
(3) **R** *is a chordal, conformal hypergraph.*
(4) *Graham's algorithm succeeds with input* **R**.
(5) *The join dependency* $\bowtie$**R** *is equivalent to a set of multivalued dependencies.*
(6) *The join dependency* $\bowtie$**R** *is equivalent to a conflict-free set of multivalued dependencies.*
(7) *Every pairwise consistent database over* **R** *is globally consistent.*
(8) *Every database over* **R** *has a full reducer.*
(9) **R** *has a join tree.*
(10) **R** *has the running intersection property.*
(11) **R** *has a monotone join expression.*
(12) **R** *has a monotone, sequential join expression.*

4. *Significance of Results and Relationship to Other Work*

Before anyone considered the question as to when a join dependency is equivalent to a set of multivalued dependencies (condition (5) of Theorem 3.4), the converse question was studied, as to which sets of multivalued dependencies are equivalent to a single join dependency. The desirability of such sets of multivalued dependencies was discussed by Sciore [36]. The points made there depend on the detailed analysis of the way the schematic notion of "objects" interacts with multivalued dependencies and with the insertion and deletion of information in the database, which analysis was done by Sciore [35]. The notion of "conflict freedom" [26] is another attempt to put restrictions on sets of multivalued dependencies in order to avoid problems in defining how the database is to be updated, and also is an attempt to establish the equivalence between relational descriptions of the real world and descriptions in more "classical" terms, such as Bachman diagrams [3, 26, 43], which are certain

directed graphs on collections of attributes. Conflict-free sets of multivalued dependencies have several nice properties: (1) they allow a unique fourth-normal-form [15] decomposition, and (2) all multivalued dependencies participate in the decomposition process; that is, the phenomenon where decomposing according to one multivalued dependency prevents another multivalued dependency from being applied does not occur.

Conditions (8) and (9) concerning 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. Semijoins can be used to help cut down on the amount of communication required in taking a join of a collection of relations at distinct sites.

Condition (7), that pairwise and global consistency be the same, was originally considered as a way of testing whether a database is the projection of a universal relation. The equivalence of (1) and (7) also says that if the relations of the database satisfy an acyclic join dependency, then we can maintain a universal relation, of which each database relation is a 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 [25, 27, 29, 35, 41, 42]. When inserting or deleting from some relation r_i, we adjust the universal relation by considering interactions among the tuples of r_i and the other relations, where we insert tuples with nulls into these relations only when necessary. In the more general (not necessary acyclic) case, the problem of adjusting the relations to maintain the property that they are the projection of a universal relation is NP-complete [24].

Condition (4) gives a polynomial-time algorithm for testing the acyclicity property. A linear-time algorithm has recently been given by Tarjan and Yannakakis [38]. Condition (10), the running intersection property, is a convenient tool for proving properties of acyclic hypergraphs. Conditions (11) and (12) are of interest because of the space-efficiency of monotone join expressions. Monotone join expressions (as in condition (11)) guarantee that no intermediate join has more tuples than the final join. Monotone, *sequential* join expressions (as in condition (12)) have the further advantage that only one intermediate join needs to be maintained at a time.

The equivalence of (1)–(4), (9), and (10) is an interesting graph-theoretic fact in its own right.

Some of the implications of Theorem 3.4 were shown previously by others. In particular, the equivalence of (1) and (5) was shown by Fagin et al. [17], the equivalence of (8) and (9) by Bernstein and Goodman [12], the equivalence of (4) and (8) by Yu and Ozsoyoglu [44], and the equivalence of (7) and (8) by Honeyman [23]. Graham [21] showed that (4) implies (7). By making use of results of Mendelzon and Maier [31] and Beeri and Vardi [9], it follows easily that (4) implies that $\bowtie\mathbf{R}$ is equivalent to a set of *embedded* multivalued dependencies [15], which is a slightly weaker statement than (5).

Other implications of Theorem 3.4 were shown independently by others. In particular, Goodman and Shmueli [19] have independently shown the equivalence of (7) and (9). Further, they give [19] a characterization of (9) from which they easily show [20] independently the equivalence of (3) and (9).

This paper makes several contributions. (1) By using hypergraphs, we unify three distinct areas of relational database theory, involving (a) relation schemes, (b) dependencies, and (c) query processing (see the introduction for more discussion); (2) we present new equivalences among previously studied concepts; and (3) we give much simpler proofs for some previously known equivalences.

FIGURE 5 

5. *Other Types of Acyclicity for Hypergraphs*

Several other types of acyclicity, none equivalent to our definition, have been defined for hypergraphs. All but one of these other types of acyclicity are more restrictive than our definition. Each of them coincides with the usual definition of acyclicity when we restrict our attention to ordinary undirected graphs. That is, an ordinary graph is acyclic in the usual sense if and only if it is acyclic in any or all of these hypergraph senses, when it is considered as a hypergraph.

We begin with the classic definition, which is due to Berge [10] and is the most restrictive type of acyclicity for hypergraphs.

A *Berge cycle* in a hypergraph $\mathscr{H}$ is a sequence $(S_1, x_1, S_2, x_2, \ldots, S_m, x_m, S_{m+1})$ such that

(i) $x_1, \ldots, x_m$ are distinct nodes of $\mathscr{H}$;
(ii) $S_1, \ldots, S_m$ are distinct edges of $\mathscr{H}$, and $S_{m+1} = S_1$;
(iii) $m \geq 2$, that is, there are at least two edges involved; and
(iv) x_i is in S_i and S_{i+1} $(1 \leq i \leq m)$.

A hypergraph is *Berge-cyclic* if it has a Berge cycle; otherwise, it is *Berge-acyclic*. The hypergraph of Figure 5, with edges ABC and BCD, is Berge-cyclic, because it contains the Berge cycle $(\underline{ABC}, C, \underline{BCD}, B, \underline{ABC})$, where, for clarity, we have underlined the edges. However, this hypergraph is acyclic under our definition. As we see by this example, if the hypergraph contains a pair of edges with more than one node in common, then the hypergraph is Berge-cyclic. A restriction that no two relation schemes can have more than one attribute in common is far too severe. Hence, Berge-acyclicity is too restrictive an assumption to make about database schemes.

Zaniolo [45] defined two types of acylicity for hypergraphs in a pioneering effort to find a condition on a hypergraph **R** that is equivalent to a certain desirable database condition ("every pairwise consistent database over **R** is globally consistent"; see the discussion of condition 3.7 above). Unfortunately, one of his conditions was sufficient but not necessary, and the other was necessary but not sufficient. Of course, our definition of acyclicity is both necessary and sufficient. The second of Zaniolo's definitions gives the only type of acyclicity that has been defined in the hypergraph literature that is less restrictive than ours.

Graham [21] weakened Zaniolo's first definition of acyclicity in another attempt to find a condition on a hypergraph **R** equivalent to "every pairwise consistent database over **R** is globally consistent." Like Zaniolo's first definition, Graham's condition was sufficient but not necessary.

Fagin [16] has recently defined two types of acyclicity for hypergraphs, which he calls β-acyclicity and γ-acyclicity (where our type of acyclicity he calls α-acyclicity). A hypergraph is β-acyclic if and only if every subset of its edges forms a hypergraph that is acyclic in our sense. Thus, for each of the various desirable properties $\mathscr{P}$ that we show are equivalent to acyclicity for database schemes, it then follows that a database scheme is β-acyclic if and only every subscheme enjoys property $\mathscr{P}$. (A subscheme of a database scheme is a subset of the relation schemes.) If turns out that β-acyclicity is equivalent to Graham's condition. Fagin's γ-acyclicity is even more restrictive than β-acyclicity. He shows that γ-acyclicity is equivalent to several desirable database conditions involving monotone-increasing joins and unique rela-

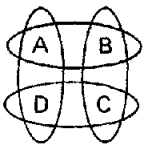


FIGURE 6

tionships among attributes. A hypergraph is γ-acyclic if and only its Bachman diagram [3] is loop-free [26, 43].

We also note that Batini et al. [4] discuss the issue of generating various subclasses of acyclic hypergraphs by "hypergraph grammars."

6. *Proof of Main Theorem*

In this section we prove our main result, Theorem 3.4, which we repeat below. We begin with a definition and a useful lemma.

Let $(\mathcal{N}, \mathcal{E})$ be a hypergraph, and let $\mathcal{F}$ be a subset of the set $\mathcal{E}$ of edges. Let $\mathcal{M}$ be the set of nodes that is the union of the members of $\mathcal{F}$. We say that $\mathcal{F}$ is *guarded* if there is an edge F (called the *guard*) in $\mathcal{F}$ such that for each edge E of the hypergraph that is not in $\mathcal{F}$, we have $E \cap \mathcal{M} \subseteq F$. Recall that we say that $\mathcal{F}$ is *closed* if for each edge E of the hypergraph there is an edge F in $\mathcal{F}$ such that $E \cap \mathcal{M} \subseteq F$. It follows easily that every guarded set of edges is closed. The converse is false. For example, consider the hypergraph in Figure 6, with edges $\{AB, BC, CD, DA\}$. The set $\{AB, BC\}$ of edges is closed but not guarded.

LEMMA 6.1. *Let $\mathcal{F}$ be a guarded set of edges of a hypergraph. An articulation set for $\mathcal{F}$ is an articulation set for the entire hypergraph.*

Note. The lemma is false if we replace "guarded" by "closed." For example, B is an articulation set for the subset $\mathcal{F} = \{AB, BC\}$ of the hypergraph in Figure 6 but not for the whole hypergraph. As we noted, $\mathcal{F}$ is closed but not guarded.

PROOF. Let (E, F) be an articulation pair for the guarded set $\mathcal{F}$ of edges of hypergraph $\mathcal{H}$. We shall show that (E, F) is an articulation pair for the whole hypergraph $\mathcal{H}$. Let $\mathcal{C}$ be a connected component of $\mathcal{F}$ that is split into at least two connected components after articulation by $Q = E \cap F$. Let $\mathcal{C}_1$ and $\mathcal{C}_2$ be two nonempty disjoint subsets of $\mathcal{C}$ such that the reductions of $\mathcal{C}_i - Q = \{C - Q : C \in \mathcal{C}_i\}$ $(i = 1, 2)$ are connected components of $\mathcal{C} - Q$. Thus, if T_1 and T_2 are arbitrary members of $\mathcal{C}_1$ and $\mathcal{C}_2$, respectively, then there is no sequence $X_1, \ldots, X_t$ of members of $\mathcal{F}$ such that

(a) $T_1 = X_1$,
(b) $T_2 = X_t$, and
(c) $X_i \cap X_{i+1} - Q$ is nonempty, for $1 \leq i \leq t$.

We know that $\mathcal{C}$ is part of a connected component of the whole hypergraph. To prove the lemma, it is sufficient to show that $\mathcal{C}_1$ and $\mathcal{C}_2$ are subsets of distinct connected components of $\mathcal{H}$ after articulation by Q. That is, it is sufficient to show that there is no sequence $X_1, \ldots, X_t$ of members of $\mathcal{H}$ such that for some T_1 in $\mathcal{C}_1$ and T_2 in $\mathcal{C}_2$, each of (a)–(c) above hold.

Assume not. Then (a)–(c) hold for appropriate choices of T_1, T_2, and $X_1, \ldots, X_t$. We know that some X_i is not in $\mathcal{F}$, since by assumption, (a)–(c) are false if every X_i is in $\mathcal{F}$. Let u be the minimum value of i and v the maximum value of i such that X_i is not in $\mathcal{F}$. Then $1 < u \leq v < t$. Denote by F the guard of the guarded set $\mathcal{F}$ of edges. Consider the sequence of edges $X_1, X_2, \ldots, X_{u-1}, F, X_{v+1}, \ldots, X_t$, in which we have

"spliced" the guard F in place of $X_u, \ldots, X_v$. Since each edge in this sequence is in $\mathscr{F}$, we can derive a contradiction by showing that this sequence of edges is one in which each consecutive pair has a node in common that is not in Q. We already know that $X_i \cap X_{i+1}$ has a node that is not in Q $(1 \leq i < t)$. So we need only show that $X_{u-1} \cap F$ and $F \cap X_{v+1}$ each have a node that is not in Q. Now $X_{u-1} \cap X_u$ has a node A that is not in Q, by assumption. Since X_{u-1} is in $\mathscr{F}$ and X_u is not, and since $\mathscr{F}$ is guarded, with guard F, it follows that $A \in F$. Hence, $X_{u-1} \cap F$ has a node (namely, A) that is not in Q, and similarly for $F \cap X_{v+1}$. This contradiction completes the proof. □

THEOREM 3.4. *The following conditions on* **R** *are equivalent:*

(1) **R** *is an acyclic hypergraph.*
(2) **R** *is a closed-acyclic hypergraph.*
(3) **R** *is a chordal, conformal hypergraph.*
(4) *Graham's algorithm succeeds with input* **R**.
(5) *The join dependency* $\bowtie$**R** *is equivalent to a set of multivalued dependencies.*
(6) *The join dependency* $\bowtie$**R** *is equivalent to a conflict-free set of multivalued dependencies.*
(7) *Every pairwise consistent database over* **R** *is globally consistent.*
(8) *Every database over* **R** *has a full reducer.*
(9) **R** *has a join tree.*
(10) **R** *has the running intersection property.*
(11) **R** *has a monotone join expression.*
(12) **R** *has a monotone, sequential join expression.*

PROOF. We shall neglect condition (6) until Section 8. We now prove the equivalence of the other conditions. We shall show that (4) $\Rightarrow$ (3) $\Rightarrow$ (4) $\Rightarrow$ (9) $\Rightarrow$ (10) $\Rightarrow$ (8) $\Rightarrow$ (7) $\Rightarrow$ (2) $\Rightarrow$ (5) $\Rightarrow$ (1) $\Rightarrow$ (2) $\Rightarrow$ (4), which shows that conditions (1)–(5) and (7)–(10) are all equivalent, and then (10) $\Rightarrow$ (12) $\Rightarrow$ (11) $\Rightarrow$ (7), which shows that (11) and (12) are equivalent to each of these. It is an instructive exercise for the reader to prove for himself directly some of the other implications.

(4) $\Rightarrow$ (3): Assume that Graham's algorithm succeeds with input **R**. Recall that Graham's algorithm [21] applies the following two operations to $\mathbf{R} = \{R_1, \ldots, R_n\}$ repeatedly until neither can be applied:

(a) If A is an attribute that appears in exactly one R_i, then delete A from R_i.
(b) Delete one R_i if there is an R_j with $j \neq i$ such that $R_i \subseteq R_j$.

Since Graham's algorithm succeeds with input **R**, this means that with input **R** the algorithm terminates with the empty set. We shall show that **R** is a chordal, conformal hypergraph. Let us denote the hypergraph **R** by $\mathscr{H}$. Let $G = G(\mathscr{H})$ be the graph of the hypergraph $\mathscr{H}$. Recall that this means that the nodes of G are the nodes of $\mathscr{H}$ and that there is an edge between two nodes of G precisely if they both lie in some hyperedge of $\mathscr{H}$.

We first show that $\mathscr{H}$ is chordal, that is, that its graph $G(\mathscr{H})$ is chordal. Suppose that $G(\mathscr{H})$ contains a chordless cycle C with at least four nodes. Since Graham's algorithm succeeds, it is easy to see that every node is eliminated by an application of rule (a) of Graham's algorithm (namely, when it is deleted for the very last time). Let v be the node of the chordless cycle C that is first eliminated by an application of rule (a) of Graham's algorithm, and let x, y be the nodes of C adjacent to v. Since v belongs to only one hyperedge of $\mathscr{H}$ when it is eliminated, this hyperedge must

contain x and y. Therefore, $G(\mathscr{H})$ contains an edge (x, y). This contradicts the fact that C was assumed to be chordless.

We now show that $\mathscr{H}$ is conformal; that is, for every clique V in $G(\mathscr{H})$ there is a hyperedge of $\mathscr{H}$ that contains V. Let V be a clique in $G(\mathscr{H})$, and let v be the node of V that is first eliminated by an application of rule (a) of Graham's algorithm. Since v belongs to only one hyperedge of $\mathscr{H}$ when it is eliminated, the hyperedge must contain V.

(3) $\Rightarrow$ (4): Assume that $\mathbf{R}$ is a chordal, conformal hypergraph. We shall show that Graham's algorithm succeeds with input $\mathbf{R}$. Let us denote the hypergraph $\mathbf{R}$ by $\mathscr{H}$.

Since $G = G(\mathscr{H})$ is chordal, it contains a simplicial node v, as noted in Section 3. That is, v together with all of its neighbors in G forms a clique V in G (a *neighbor* of v in G is a node w such that (v, w) is an edge of G). Since $\mathscr{H}$ is conformal, there is a hyperedge W of $\mathscr{H}$ that contains V. Let X be an arbitrary hyperedge of $\mathscr{H}$ that contains v as a member. By construction of G, we know that $X \subseteq V$ (for, if w is an arbitrary node other than v in X, then (v, w) is an edge of G, and so $w \in V$). Since also $V \subseteq W$, it follows that $X \subseteq W$. Thus every hyperedge X of $\mathscr{H}$ that contains v is a subset of W. So, by applications of rule (b) of Graham's algorithm, we are left with a hypergraph in which there is only one hyperedge (namely, W) of $\mathscr{H}$ that contains v. By an application of rule (a) of Graham's algorithm, node v is then deleted, since it appears in only one hyperedge.

It is easy to verify that the hypergraph that remains after applying a step of Graham's algorithm to a chordal, conformal hypergraph yields a chordal, conformal hypergraph. Thus it is possible to proceed inductively by selecting a simplicial node for the remaining hypergraph. In this way all nodes are eventually deleted, and so Graham's algorithm succeeds with input $\mathbf{R}$.

(4) $\Rightarrow$ (9): Assume that Graham's algorithm succeeds with input $\mathbf{R}$. We shall show that $\mathbf{R}$ has a join tree. We build a join tree T for $\mathbf{R}$ as follows. We take the members $R_1, \ldots, R_n$ of $\mathbf{R}$ as nodes of T. Run Graham's algorithm on $\mathbf{R}$. At the end of the mth step of the algorithm (where a *step* consists of an application of one of rules (a) or (b) as described above), denote by $\text{rem}_m(R_i)$ what is left of R_i. If $\text{rem}_m(R_i)$ is empty but $\text{rem}_{m-1}(R_i)$ is not, then we say that "R_i is deleted on the mth step." On any given step at most one R_i is deleted. If R_i is deleted on the mth step because $\text{rem}_{m-1}(R_i) \subseteq \text{rem}_{m-1}(R_j)$ and because rule (b) of Graham's algorithm was applied on the mth step, then add edge (R_i, R_j) to T with label $R_i \cap R_j$. (If there are several such j's, then arbitrarily select just one of them.) We say that R_i is a *child* of R_j and R_j is the *parent* of R_i. Obviously, R_j is deleted on a later step than R_i is. It is clear that we obtain a forest (a collection of trees) in this manner. By adding some edges we can convert the forest into a tree. (We simply add just enough edges, chosen arbitrarily, to "connect" the trees in the forest into a single tree.) We now show that the resulting tree T is a join tree.

If T is not a join tree, then there are i and j $(i \neq j)$ and a node A such that

(i) $A \in R_i \cap R_j$, and
(ii) the path in T between R_i and R_j is not A-labeled, that is, some edge along the path does not have label A (possibly among others).

Choose R_i and R_j so that (i) and (ii) hold and the earlier of the times that R_i or R_j was deleted is as late as possible. Assume without loss of generality that R_i was deleted before R_j. Since R_i and R_j have a node (namely, A) in common, we know that R_i was deleted by an application of rule (b) of Graham's algorithm and not by

an application of rule (a). Thus there is some k ($k \neq i$) such that $\text{rem}_{m-1}(R_i) \subseteq \text{rem}_{m-1}(R_k)$, where m is the step on which R_i is deleted and R_k is the parent of R_i in the tree T. So R_k was deleted on a later step than R_i. Now R_k and R_j both contain node A, and so, by our maximality assumption in our choice of i and j, we know that either $k = j$ or the path in T between R_k and R_j is A-labeled. In the latter case this path can be extended to an A-labeled path between R_i and R_j, since R_i is a child of R_k. In the former case the edge between R_i and R_j is an A-labeled path between R_i and R_j. So in either case the path between R_i and R_j is A-labeled. This is a contradiction.

(9) $\Rightarrow$ (10): Assume that $\mathbf{R}$ has a join tree. We shall show that $\mathbf{R}$ has the running intersection property. Recall that we say that $\mathbf{R}$ has the *running intersection property* if there is an ordering $R_1, \ldots, R_n$ of $\mathbf{R}$ such that for $2 \leq i \leq n$ there exists $j_i < i$ such that $R_i \cap (R_1 \cup \cdots \cup R_{i-1}) \subseteq R_{j_i}$. That is, the intersection of each R_i with the union of the previous R_j's is contained in one of these.

Let T be a join tree for $\mathbf{R}$. Select a root for the tree T. Let $R_1, \ldots, R_n$ be an ordering of $\mathbf{R}$ by increasing depth. Thus, if R_j is the parent of R_i, then $j < i$. Clearly, each path from R_i to any of $R_1, \ldots, R_{i-1}$ must pass through R_i's parent R_j. Now if A is a node in $R_i \cap R_k$ for some $k < i$, then the path between R_i and R_k is A-labeled. Since this path passes through R_j, it follows that $A \in R_j$. It follows that $R_j \cap (R_1 \cup \cdots \cup R_{i-1}) \subseteq R_j$. Thus j is the j_i demanded in the definition of the running intersection property.

(10) $\Rightarrow$ (8): Assume that $\mathbf{R}$ has the running intersection property. We now give a semijoin program (which is modeled after one by Bernstein and Chiu [11]) that we shall prove is a full reducer for $\mathbf{R}$, that is, which converts the relations $r_1, \ldots, r_n$ into new relations $r'_1, \ldots, r'_n$ that are globally consistent. (Recall that a set of relations is *globally consistent* if there is a universal relation such that each is the appropriate projection of the universal relation. Further, as we noted, if there is any such universal relation, then the join of the set of relations is such a universal relation.) Let $R_1, \ldots, R_n$ be an ordering of $\mathbf{R}$ as guaranteed by the running intersection property. Thus, for $2 \leq i \leq n$ there exists $j_i < i$ such that $R_i \cap (R_1 \cup \cdots \cup R_{i-1}) \subseteq R_{j_i}$. For later reference we have labeled the lines of the program, some labels with negative integers and some with positive integers:

$(-n)$ $\quad r_{j_n} := r_{j_n} \ltimes r_n$

$(-n+1)$ $\quad r_{j_{n-1}} := r_{j_{n-1}} \ltimes r_{n-1}$

$\vdots$

$(-i)$ $\quad r_{j_i} := r_{j_i} \ltimes r_i$

$\vdots$

(-2) $\quad r_{j_2} := r_{j_2} \ltimes r_2$

(2) $\quad r_2 := r_2 \ltimes r_{j_2}$

$\vdots$

(i) $\quad r_i := r_i \ltimes r_{j_i}$

$\vdots$

(n) $\quad r_n := r_n \ltimes r_{j_n}$

Note that $j_2 = 1$ in lines (-2) and (2) above.

Let us denote by $r'_1, \ldots, r'_n$ the result of the program, starting with input $r_1, \ldots, r_n$. We shall show that $r'_1, \ldots, r'_n$ are globally consistent. We begin by proving that r'_i and r'_{j_i} are consistent for each i $(2 \leq i \leq n)$, that is, that $r'_i[R_i \cap R_{j_i}] = r'_{j_i}[R_i \cap R_{j_i}]$ for each i $(2 \leq i \leq n)$. Let us write j for j_i and Q for $R_i \cap R_j$. Thus, to show that r'_i and r'_j are consistent, we must show that $r'_i[Q] = r'_j[Q]$.

Let us denote by $r_k^{(p)}$ the "current value" of the relation with attributes R_k immediately after line (p) of the program is executed (where $1 \leq k \leq n$ and (p) is a line number of the program). In particular, $r'_k = r_k^{(n)}$ for each k.

In what follows we shall frequently make tacit use of the following two simple facts about semijoins (where r and s are relations with attributes R and S, respectively): $(r \ltimes s) \subseteq r$, and $(r \ltimes s)[R \cap S] \subseteq s[R \cap S]$.

From line $(-i)$ of the semijoin program, we see that

$$r_j^{(-i)}[Q] \subseteq r_i^{(-i)}[Q]. \tag{6.1}$$

In those lines of the program strictly between lines $(-i)$ and (i) the expression r_i never appears on the left-hand side of an assignment. For, if line $(-k)$ is one of the negatively numbered lines with $k < i$, then the left-hand side of the assignment is r_{j_k}, and $j_k < k < i$; and if line (k) is a positively numbered line with $k < i$, then the left-hand side of the assignment is r_k, and $k < i$. So,

$$r_i^{(-i)} = r_i^{(i-1)}. \tag{6.2}$$

From (6.1) and (6.2) it follows that

$$r_j^{(-i)}[Q] \subseteq r_i^{(i-1)}[Q]. \tag{6.3}$$

Because relations can only lose, never gain, tuples in a semijoin program, it is easy to see that for every k, p, q such that $p \leq q$, necessarily $r_k^{(q)} \subseteq r_k^{(p)}$. In particular,

$$r_j^{(i-1)} \subseteq r_j^{(-i)}.$$

Hence,

$$r_j^{(i-1)}[Q] \subseteq r_j^{(-i)}[Q]. \tag{6.4}$$

From (6.4) and (6.3) and transitivity of set inclusion it follows that

$$r_j^{(i-1)}[Q] \subseteq r_i^{(i-1)}[Q]. \tag{6.5}$$

Because of (6.5), an application of line (i) of the semijoin program causes

$$r_i^{(i)}[Q] = r_j^{(i)}[Q]. \tag{6.6}$$

Since no line (p) with $p > i$ has either r_i or r_j on the left-hand side of an assignment (because $j < i < p$), it follows that $r_i^{(n)} = r_i^{(i)}$ and $r_j^{(n)} = r_j^{(i)}$ From this fact and from (6.6) we find that

$$r_i^{(n)}[Q] = r_j^{(n)}[Q]. \tag{6.7}$$

But (6.7) simply says that $r'_i[Q] = r'_j[Q]$, which was to be shown.

We have shown that for each i $(2 \leq i \leq n)$, the relations r'_i and r'_{j_i} are consistent. We now show that from this fact and the fact that $R_i \cap (R_1 \cup \cdots \cup R_{i-1}) \subseteq R_{j_i}$ for each i $(2 \leq i \leq n)$, it follows that $r'_1, \ldots, r'_n$ are globally consistent. Define q_k $(1 \leq k \leq n)$ to be $r'_1 \bowtie \cdots \bowtie r'_k$. We shall prove by induction on k $(1 \leq k \leq n)$ that

$$r'_i = q_k[R_i] \qquad \text{for} \quad 1 \leq i \leq k. \tag{6.8}$$

The $k = 1$ case is trivial. Assume that (6.8) is true for k $(1 \leq k < n)$; we shall show

it for $k + 1$. That is, we shall show that

$$r'_i = q_{k+1}[R_i] \qquad \text{for} \quad 1 \le i \le k + 1. \tag{6.9}$$

Let $V = R_{k+1} \cap (R_1 \cup \cdots \cup R_k)$, and let $j = j_{k+1}$. By definition of $j = j_{k+1}$ we know that $V = R_{k+1} \cap R_j$. We have shown that r'_{k+1} and r'_j are consistent, that is, that

$$r'_{k+1}[V] = r'_j[V]. \tag{6.10}$$

Now by (6.8) and the fact that $j = j_{k+1} < k + 1$, it follows that $r'_j = q_k[R_j]$. So, since $V \subseteq R_j$, we know that $r'_j[V] = q_k[V]$. This fact, along with (6.10), implies that $r'_{k+1}[V] = q_k[V]$. Hence r'_{k+1} and q_k are consistent. Therefore $r'_{k+1} = (r'_{k+1} \bowtie q_k)[R_{k+1}]$. But $r'_{k+1} \bowtie q_k = q_{k+1}$. Hence,

$$r'_{k+1} = q_{k+1}[R_{k+1}]. \tag{6.11}$$

This proves (6.9) when $i = k + 1$.

We now prove (6.9) when $1 \le i \le k$. By (6.8) we know that $r'_i = q_k[R_i]$. By consistency of r'_{k+1} and q_k (which we showed above), it follows that $q_k[R_i]$ equals $(r'_{k+1} \bowtie q_k)[R_i]$, which in turn equals $q_{k+1}[R_i]$. Putting together the equalities we have shown in this paragraph, it follows that

$$r'_i = q_{k+1}[R_i] \qquad \text{for} \quad 1 \le i \le k. \tag{6.12}$$

Now (6.11) and (6.12) give us (6.9), which completes the induction step. Hence (6.8) holds for each k ($1 \le k \le n$), and, in particular, when $k = n$. So $r'_1, \ldots, r'_n$ are globally consistent, since they are each projections of q_n. Thus **R** has a full reducer, which was to be shown.

(8) $\Rightarrow$ (7): Assume that every database over **R** has a full reducer. We shall show that every pairwise consistent database over **R** is globally consistent. Let $\mathbf{r} = \{r_1, \ldots, r_n\}$ be a pairwise consistent database over **R**. We must show that **r** is globally consistent. By assumption, **r** has a full reducer. However, the input and output to this semijoin program (the full reducer) are the same, by pairwise consistency. But we are guaranteed that the output of the full reducer is a globally consistent database. So **r** is globally consistent, which was to be shown.

(7) $\Rightarrow$ (2): We must show that if every pairwise consistent database over **R** is globally consistent, then **R** is a closed-acyclic hypergraph. If this implication is false, then let $\mathbf{R} = \{R_1, \ldots, R_n\}$ be a counterexample with n as small as possible and such that, relative to n, the number of attributes is minimized. By minimality, it follows easily that **R** is necessarily connected.

We first show that the Graham algorithm leaves **R** unchanged. That is, we shall show that

(a) $R_i \subseteq \bigcup \{R_j : j \ne i\}$ for each i. In other words, each attribute is in at least two R_i's.
(b) $R_i \not\subseteq R_j$ if $i \ne j$. In other words, no R_i is a subset of any other R_j.

Assume that either (a) or (b) were false. Then we could apply one step of Graham's algorithm to obtain $\mathbf{R}'$ from **R**, wherein either an attribute that appears in only one R_i is deleted or else an R_i that is a subset of a different R_j is deleted. It is simple to see that because every pairwise consistent database over **R** is globally consistent, it follows that every pairwise consistent database over $\mathbf{R}'$ is globally consistent. We now show that $\mathbf{R}'$ is not a closed-acyclic hypergraph, which contradicts our assumed minimality of **R**. If $\mathbf{R}'$ were obtained from **R** by deleting an attribute that appears in only one R_i, then it is easy to see that a nontrivial, connected, closed set of edges with

A_1	A_2	$\cdots$	A_i	$\cdots$	A_p	A_{p+1}	A_{p+2}	$\cdots$	A_m
1	0	$\cdots$	0	$\cdots$	0	1	1	$\cdots$	1
0	1	$\cdots$	0	$\cdots$	0	2	2	$\cdots$	2
$\vdots$	$\vdots$		$\vdots$		$\vdots$	$\vdots$	$\vdots$		$\vdots$
0	0	$\cdots$	1	$\cdots$	0	i	i	$\cdots$	i
$\vdots$	$\vdots$		$\vdots$		$\vdots$	$\vdots$	$\vdots$		$\vdots$
0	0	$\cdots$	0	$\cdots$	1	p	p	$\cdots$	p

FIGURE 7

no articulation set in **R** (which exists since **R** is not closed-acyclic) immediately gives us the same in **R'**. If **R'** were obtained from **R** by deleting an R_i that is a subset of a different R_j, then **R'** and **R** would have the same reduction. So **R'** is not closed-acyclic, because **R** is not (recall that we defined a hypergraph to be closed-acyclic precisely if its reduction is). This contradicts our assumed minimality of **R**. Hence (a) and (b) above hold.

There are now two cases, depending on whether or not $R_2 - R_1$, $R_3 - R_1$, ..., $R_n - R_1$ are connected.

Case 1. $R_2 - R_1$, $R_3 - R_1$, ..., $R_n - R_1$ are connected. Assume that R_1 has attributes $A_1, A_2, \ldots, A_p$, and that $A_{p+1}, \ldots, A_m$ are the other attributes $\mathcal{N} - R_1$ (where $\mathcal{N}$ is the set of all attributes). Let relation r be as in Figure 7. There are p tuples $w_1, \ldots, w_p$. Tuple w_i has 1 in column A_i, 0 in the other columns of R_1, and i in the columns of $\mathcal{N} - R_1$. Let $r_i = r[R_i]$, for $1 \leq i \leq n$. We now show that

$$r = r_2 \bowtie r_3 \bowtie \cdots \bowtie r_n. \tag{6.13}$$

Note that the right-hand side of the equality in (6.13) is a relation over all of the attributes, by (a) above, and so (6.13) at least makes sense. The inclusion

$$r \subseteq r_2 \bowtie r_3 \bowtie \cdots \bowtie r_n$$

is automatic, since each r_i is a projection of r. We now prove the opposite inclusion, that is, that

$$r_2 \bowtie \cdots \bowtie r_n \subseteq r. \tag{6.14}$$

Let u be a tuple in $r_2 \bowtie \cdots \bowtie r_n$; we must show that u is a tuple in r. Since u is in $r_2 \bowtie \cdots \bowtie r_n$, we know that $u[R_i]$ is in r_i for $2 \leq i \leq n$. But $r_i = r[R_i]$, so $u[R_i]$ is in $r[R_i]$, for $2 \leq i \leq n$. This means that there is a tuple q_i of r such that $u[R_i] = q_i[R_i]$, for $2 \leq i \leq n$. We shall show that all of the q_i's are equal. It then follows that u equals their common value, since by (a) above, all of the attributes appear in $\bigcup\{R_j : j \neq 1\}$. This implies that u is in r, which completes the proof of (6.14), and hence of (6.13).

Thus, to prove (6.13), we need only show that all of the q_i's are equal. If they are not all equal, then find j and k such that $q_j \neq q_k$. By assumption, $R_2 - R_1$, $R_3 - R_1$, ..., $R_n - R_1$ are connected. Hence there is a sequence $i_1, \ldots, i_m$ of integers, each in $\{2, \ldots, n\}$, so that

(i) $i_1 = j$,
(ii) $i_m = k$, and
(iii) $R_{i_h} - R_1$ and $R_{i_{h+1}} - R_1$ have in common an attribute B_h $(1 \leq h < m)$.

Let B_h be as in (iii). Then $q_{i_h}[B_h] = u[B_h] = q_{i_{h+1}}[B_h]$, for $1 \leq h < m$. So q_{i_h} and $q_{i_{h+1}}$ are two tuples of r that agree on an attribute, namely, B_h, that is not in

R_1. Therefore, q_{i_h} and $q_{i_{h+1}}$ are equal ($1 \leq h < u$), as we see by the definition of r. Hence, all of the q_{i_h}'s are equal, and, in particular, $q_{i_1} = q_{i_m}$. But $q_{i_1} = q_j \neq q_k = q_{i_m}$. This contradiction completes the proof of (6.13).

Let us define r_1' (with attributes R_1) to contain the tuples of r_1, along with a new tuple of all 0's. If $2 \leq i \leq n$, then $R_1 \cap R_i$ is a proper subset of R_1, by condition (b) above. So, it is easy to see that $r_1'[R_1 \cap R_i] = r_1[R_1 \cap R_i]$, for $2 \leq i \leq n$. Hence, $\mathbf{r}' = \{r_1', r_2, \ldots, r_n\}$ is pairwise consistent, because $\mathbf{r} = \{r_1, r_2, \ldots, r_n\}$ is pairwise consistent (in fact, $\mathbf{r}$ is even globally consistent, since each member of $\mathbf{r}$ is a projection of r). However, we now show that $\mathbf{r}'$ is not globally consistent. If $\mathbf{r}'$ were globally consistent, then, as we noted in the discussion following Condition 3.7, each member of $\mathbf{r}'$ would be a projection of $r_1' \bowtie r_2 \bowtie \cdots \bowtie r_n$, which equals $r_1' \bowtie (r_2 \bowtie \cdots \bowtie r_n)$, which, by (6.13), equals $r_1' \bowtie r$, which, in turn, equals r. But r_1' is not a projection of r. Thus $\mathbf{r}'$ is not globally consistent. However, we assumed that every pairwise consistent database over $\mathbf{R}$ is globally consistent. This is a contradiction.

Case 2. $R_2 - R_1, R_3 - R_1, \ldots, R_n - R_1$ are not connected. We now show that if $\mathbf{P} = \{P_1, \ldots, P_t\}$ is a closed set of edges of $\mathbf{R}$, then every pairwise consistent database over $\mathbf{P}$ is globally consistent. Let $\mathbf{p} = \{p_1, \ldots, p_t\}$ be a database over $\mathbf{P}$ that is pairwise consistent, where p_i has attributes P_i ($1 \leq i \leq t$). We must show that $\mathbf{p}$ is globally consistent. We define a database $\mathbf{r}$ over $\mathbf{R}$ that we shall show is pairwise (and hence globally) consistent. Let R_i be a member of $\mathbf{R}$. If R_i is in $\mathbf{P}$ (say $R_i = P_j$), then let r_i be p_j. If R_i is not in $\mathbf{P}$, then, since $\mathbf{P}$ is closed, we know that there is a member E of $\mathbf{P}$ such that $R_i \cap \mathscr{P} \subseteq E$, where $\mathscr{P}$ is the set of nodes in $\mathbf{P}$. We define r_i by letting $r_i[R_i \cap E]$ be $e[R_i \cap E]$ (where e is the member of $\mathbf{P}$ with attributes E) and letting all other entries (in the other columns) in every tuple in r_i be 0. We now show that $\mathbf{r}$ is pairwise consistent. Let R_i and R_j be two distinct members of $\mathbf{R}$. We shall show that r_i and r_j are consistent. There are three possibilities.

(a) Assume that R_i and R_j are both in $\mathbf{P}$. The consistency of the corresponding relations r_i and r_j follows from the pairwise consistency of $\mathbf{p}$.

(b) Assume that R_i is not in $\mathbf{P}$ but R_j is. Let E and e be as above. Denote $R_i \cap R_j$ by Q. Since R_j is in $\mathbf{P}$, it follows by definition of E that $Q \subseteq (R_i \cap E)$. So, since r_i and e are consistent by construction, it follows that

$$r_i[Q] = e[Q]. \tag{6.15}$$

We already saw that $Q \subseteq (R_i \cap E) \subseteq E$. Hence, since $Q \subseteq R_j$, it follows that $Q \subseteq (R_j \cap E)$. Now r_j and e are consistent (because both are in the pairwise consistent database $\mathbf{p}$). Therefore,

$$r_j[Q] = e[Q]. \tag{6.16}$$

It follows from (6.15) and (6.16) that $r_i[Q] = r_j[Q]$. Hence, r_i and r_j are consistent.

(c) Assume that neither R_i nor R_j is in $\mathbf{P}$. Let E and e be as above, and let F and f be the corresponding items, when we consider R_j instead of R_i. Then $(R_i \cap R_j \cap \mathscr{P}) \subseteq (E \cap F)$. Since e and f are consistent (being members of $\mathbf{p}$), and since r_i and r_j contain only 0 entries for attributes not in $\mathscr{P}$, it follows that r_i and r_j are consistent.

Thus we have shown that $\mathbf{r}$ is pairwise consistent. Since every pairwise consistent database over $\mathbf{R}$ is globally consistent, we know that $\mathbf{r}$ is globally consistent. By restricting our attention to $\mathscr{P}$ we see that this implies that $\mathbf{p}$ is globally consistent. Hence every pairwise consistent database over $\mathbf{P}$ is globally consistent, which was to be shown.

Since every pairwise consistent database over **P** is globally consistent, for every closed subset **P** of **R**, it follows by our minimality assumption on **R** that every proper closed subset of **R** is closed-acyclic. Since **R** itself is not closed-acyclic, we know that some nontrivial, connected, closed set **P** of edges of **R** has no articulation set. But **P** cannot be a proper subset of **R**, since if it were, then it would have an articulation set (because, as we just showed, **P** is closed-acyclic). Hence **R** itself is a nontrivial, connected, closed set of edges with no articulation set.

Now by assumption, $R_2 - R_1, R_3 - R_1, \ldots, R_n - R_1$ are not connected. Let $\mathscr{F}$ be a maximal subset of $R_2, \ldots, R_n$ such that $\{F - R_1 : F \in \mathscr{F}\}$ is connected. We know that $\mathscr{F}$ is a proper subset of $\{R_2, \ldots, R_n\}$. If $\mathscr{F}$ is $\{F_1; \ldots, F_s\}$, then let $\mathscr{F}'$ be $\{F_1, \ldots, F_s, R_1\}$, the result of adding R_1 to the set $\mathscr{F}$. Since the only attributes in common between any member of $\mathscr{F}'$ and any member of **R** not in $\mathscr{F}'$ lies in R_1, which is in $\mathscr{F}'$, it follows that $\mathscr{F}'$ is a guarded set, with guard R_1. Also, $\mathscr{F}'$ contains at least two edges, since $\mathscr{F}$ contains at least one edge and $\mathscr{F}'$ also contains R_1. We now show that $\mathscr{F}'$ is connected.

Clearly $\mathscr{F}$ is connected. We now show that $\mathscr{F}'$ is connected. Assume not; we shall derive a contradiction. Since $\mathscr{F}$ is connected and $\mathscr{F}'$ is not, it is clear from the definition of $\mathscr{F}'$ that R_1 is disjoint from every member of $\mathscr{F}$. Since **R** is connected and $\mathscr{F}$ is a proper subset of **R**, it follows that there is an edge S of **R** that is not in $\mathscr{F}$ but which intersects some member E of $\mathscr{F}$. Let A be a node in $S \cap E$. Then $A \notin R_1$, since R_1 is disjoint from E. So $E - R_1$ and $S - R_1$ intersect (because both contain A). Therefore $\{F - R_1 : F \in \mathscr{F}\} \cup \{S - R_1\}$ is connected. This contradicts maximality of $\mathscr{F}$.

We have shown that $\mathscr{F}'$ is a connected, guarded set of at least two edges. Now $\mathscr{F}'$ is closed, since it is guarded. We also know that $\mathscr{F}'$ is a proper subset of **R**, since $\mathscr{F}$ is a proper subset of $R_2, \ldots, R_n$. We showed that every proper, closed subset of **R** is closed-acyclic. Hence $\mathscr{F}'$ is closed-acyclic, and so it has an articulation set. By Lemma 6.1, this articulation set is an articulation set for the whole hypergraph **R**. However, we showed that **R** has no articulation set. This is a contradiction.

(2) $\Rightarrow$ (5) $\Rightarrow$ (1) $\Rightarrow$ (2): In the proof in [17] that conditions (1) and (5) are equivalent (i.e., that **R** is an acyclic hypergraph if and only if the join dependency $\bowtie$**R** is equivalent to a set of multivalued dependencies), the proof actually showed the stronger result that if **R** is a closed-acyclic hypergraph (condition (2)), then the join dependency $\bowtie$**R** is equivalent to a set of multivalued dependencies (condition (5)), which in turn was shown to imply that **R** is an acyclic hypergraph (condition (1)). And, as noted before, the fact that acyclic implies closed-acyclic (i.e., (1) $\Rightarrow$ (2)) is almost immediate. Thus the implications (2) $\Rightarrow$ (5) $\Rightarrow$ (1) $\Rightarrow$ (2) are shown in [17].

(2) $\Rightarrow$ (4): Assume that **R** is a closed-acyclic hypergraph. We must show that Graham's algorithm succeeds with input **R**. Assume not; we shall derive a contradiction. Recall that Graham's algorithm [21] applies the following two operations to $\mathbf{R} = \{R_1, \ldots, R_n\}$ repeatedly until neither can be applied:

(a) If A is an attribute that appears in exactly one R_i, then delete A from R_i.
(b) Delete one R_i if there is an R_j with $j \neq i$ such that $R_i \subseteq R_j$.

The algorithm *succeeds* if it terminates with the empty set; otherwise, it *fails*.

We now show that if one step of Graham's algorithm (the application of one of rules (a) or (b)) is applied to a closed-acyclic hypergraph $\mathscr{H}$, then the result $\mathscr{H}'$ is a closed-acyclic hypergraph. It follows inductively (on the number of steps of Graham's algorithm that are applied) that if the input to the algorithm is a closed-acyclic hypergraph, then the algorithm terminates with a closed-acyclic hypergraph.

Assume first that rule (b) is applied. Then the hypergraphs $\mathcal{H}$ and $\mathcal{H}'$ have the same reduction. Recall that under our definition of "closed-acyclic," a hypergraph is closed-acyclic precisely if its reduction is. Thus, since $\mathcal{H}$ is closed-acyclic, and since it has the same reduction as $\mathcal{H}'$, also $\mathcal{H}'$ is closed-acyclic.

Now assume that rule (a) is applied. Let us call a node of a hypergraph *isolated* if it appears in exactly one edge. Let us denote by D the edge of $\mathcal{H}$ that contains the isolated node A that is deleted by the application of rule (a). If S is an edge of $\mathcal{H}$, then let S' be S if $S \neq D$, and $S' = D - \{A\}$ otherwise. We say that S and S' are *corresponding* edges of $\mathcal{H}$ and $\mathcal{H}'$. Let $\mathcal{F}'$ be a nontrivial, connected, closed set of edges, with no articulation set, in the reduction of $\mathcal{H}'$. To show that $\mathcal{H}'$ is closed-acyclic, we must show that $\mathcal{F}'$ has an articulation set. Let $\mathcal{F}$ be the set of corresponding edges in the reduction of $\mathcal{H}$. It is easy to see that $\mathcal{F}$ is closed, nontrivial, and connected, and hence has an articulation pair (E, F). It is straightforward (except for one subtlety) to verify that the corresponding edges E', F' of $\mathcal{F}'$ form an articulation pair for $\mathcal{F}'$. The subtlety is as follows. Assume that $E = D$, where D is the edge with the isolated node A that was just deleted. Assume that after articulation of $\mathcal{F}$ by the articulation set $E \cap F$, there are two connected components, one of which consists of the node A by itself. Then why should $E' \cap F'$ be an articulation set for $\mathcal{F}'$? The answer is as follows. Under the circumstances we have described, D' is a proper subset of F. But then D' is not an edge in the reduction of $\mathcal{H}'$. This is a contradiction, since we have assumed that $\mathcal{F}'$ is in the reduction of $\mathcal{H}'$. Thus $\mathcal{H}'$ has an articulation set $E' \cap F'$, as desired.

By assumption, the input to Graham's algorithm is a closed-acyclic hypergraph, and Graham's algorithm does not succeed. That is, the algorithm terminates with a hypergraph (call it $\mathcal{G}$) that is not empty. We just showed that because the input to Graham's algorithm is closed-acyclic, so is the output. Thus, $\mathcal{G}$ is a closed-acyclic hypergraph that is nonempty, and to which we cannot apply either of rules (a) or (b) of Graham's algorithm. We shall derive a contradiction.

Let us define a *knob* of a hypergraph to be an edge that contains an isolated node. (Recall that a node is *isolated* if it appears in exactly one edge.) We now prove inductively on the number n of edges in a hypergraph that each reduced, closed-acyclic hypergraph with at least two edges contains at least two knobs. We shall then have our contradiction. For, hypergraph $\mathcal{G}$ above is reduced and closed-acyclic. It has at least two edges: it is nonempty, and if it had only one edge, then every node would be isolated, and we could apply rule (a) of Graham's algorithm. It has no knobs, or else we could apply rule (a) of Graham's algorithm.

The basis ($n = 1$) of the induction is trivial, since it doesn't occur. For the induction step, assume that $\mathcal{H}$ is a reduced, closed-acyclic hypergraph with n edges ($n > 1$), and that every closed-acyclic hypergraph with at least two edges but less than n edges has at least two knobs. We must show that $\mathcal{H}$ has at least two knobs.

Since $\mathcal{H}$ is reduced, closed-acyclic, and has at least two edges, it has an articulation pair (E, F). Let us write the articulation set $E \cap F$ as Q. We can thus partition the edges of $\mathcal{H}$ into two disjoint, nonempty sets $\mathcal{F}_1$ and $\mathcal{F}_2$ such that whenever $F_1 \in \mathcal{F}_1$ and $F_2 \in \mathcal{F}_2$, then $F_1 \cap F_2 \subseteq Q$. There are three cases.

Case 1. $\mathcal{F}_1$ and $\mathcal{F}_2$ are both singletons. The single member of $\mathcal{F}_1$ is then a knob, as is the single member of $\mathcal{F}_2$.

Case 2. One of $\mathcal{F}_1$ and $\mathcal{F}_2$, say $\mathcal{F}_1$, is a singleton, and the other is not. The edge in $\mathcal{F}_1$ is a knob, since it contains a node not in Q and hence not in any member of $\mathcal{F}_2$. Now $\mathcal{F}_2$ contains at least one of E or F (since $\mathcal{F}_1$ is a singleton). It follows easily that $\mathcal{F}_2$ is closed. Now a closed subset of a closed-acyclic hypergraph is a closed-acyclic

hypergraph; this follows easily from the simple fact [17] that a closed subset of a closed subset is closed. Thus $\mathscr{F}_2$, considered as a hypergraph, is closed-acyclic. So by the induction hypothesis, it contains at least two knobs. We shall show that at least one of these knobs is a knob of the original hypergraph $\mathscr{H}$. There are two subcases.

Case 2a. The edge in $\mathscr{F}_1$ is one of E or F, say E. Then $F \in \mathscr{F}_2$, but $E \notin \mathscr{F}_2$. Now at least one of the knobs of $\mathscr{F}_2$ is not F. Call this knob V. Then V has a node v not in any other edge in $\mathscr{F}_2$. Hence $v \notin F$. So $v \notin Q$, and so v is not in the edge E of $\mathscr{F}_1$ (because every node in both an edge of $\mathscr{F}_1$ and an edge of $\mathscr{F}_2$ is in Q). We have shown that V is a knob of $\mathscr{H}$.

Case 2b. The edge in $\mathscr{F}_1$ is neither E nor F. So E and F are both in $\mathscr{F}_2$. Now $\mathscr{F}_2$ has two knobs V and W. Since V is a knob of $\mathscr{F}_2$, let v be a node of V that is in no other member of $\mathscr{F}_2$. Since E and F are both in $\mathscr{F}_2$, we know that v is not in $E \cap F = Q$, and so v is not in the edge of $\mathscr{F}_1$. Hence V is a knob of $\mathscr{H}$.

Case 3. $\mathscr{F}_1$ and $\mathscr{F}_2$ each have at least two edges. There are two subcases.

Case 3a. E and F are not both in the same $\mathscr{F}_i$; say $E \in \mathscr{F}_1$ and $F \in \mathscr{F}_2$. Then, as in case 2, each of $\mathscr{F}_1$ and $\mathscr{F}_2$ is closed and acyclic, and so by the induction hypothesis, each contains at least two knobs. Now at least one of the knobs of $\mathscr{F}_1$ (respectively, $\mathscr{F}_2$) is not E (respectively, F); call one such knob V_1 (respectively, V_2). By an argument almost identical to that in case 2a, it follows that V_1 and V_2 are knobs of the hypergraph $\mathscr{H}$.

Case 3b. E and F are both in the same $\mathscr{F}_i$, say $\mathscr{F}_1$. Let $\mathscr{F}'_1 = \mathscr{F}_1$, and let $\mathscr{F}'_2 = \mathscr{F}_2 \cup \{E\}$. Thus $\mathscr{F}'_2$ has all of the edges of $\mathscr{F}_2$, along with one more edge, namely, E. It is easy to see that $\mathscr{F}'_1$ and $\mathscr{F}'_2$ are closed. Hence, as before, each is a closed-acyclic hypergraph with at least two edges. It is clear that $\mathscr{F}'_1$ has strictly fewer edges than $\mathscr{H}$, since $\mathscr{F}'_1$ has none of the edges in $\mathscr{F}_2$. Further, $\mathscr{F}'_2$ has strictly fewer edges than $\mathscr{H}$, since $\mathscr{F}'_2$ does not contain the edge F. So by the induction hypothesis, $\mathscr{F}'_1$ and $\mathscr{F}'_2$ each have at least two knobs, and in particular, each has some knob that is not E. Let V_i be a knob of $\mathscr{F}'_i$, where $V_i \neq E$ $(i = 1, 2)$. Clearly $V_1 \neq V_2$, since the only edge that $\mathscr{F}'_1$ and $\mathscr{F}'_2$ have in common is E. We now show that V_1 and V_2 are each knobs of the hypergraph $\mathscr{H}$. Let v be a node in V_1 that does not appear in any other edge in $\mathscr{F}_1$. Then $v \notin E$, and so v is not in any edge of $\mathscr{F}_2$ (because every node that is in both an edge of $\mathscr{F}_1$ and an edge of $\mathscr{F}_2$ is in $E \cap F$). Thus v is an isolated node of $\mathscr{H}$, and so V_1 is a knob of $\mathscr{H}$. Similarly, V_2 is a knob of $\mathscr{H}$.

(10) $\Rightarrow$ (12): Assume that $\mathbf{R}$ has the running intersection property. Let $R_1, \ldots, R_n$ be an ordering of $\mathbf{R}$ as guaranteed by the running intersection property. Thus for $2 \leq i \leq n$ there exists $j_i < i$ such that $R_i \cap (R_1 \cup \cdots \cup R_{i-1}) \subseteq R_{j_i}$. We now show that $(\cdots ((R_1 \bowtie R_2) \bowtie R_3) \cdots \bowtie R_n)$ is a monotone, sequential join expression. That is, we shall show that if $\mathbf{r} = \{r_1, \ldots, r_n\}$ is a pairwise consistent database over $\mathbf{R} = \{R_1, \ldots, R_n\}$, then the join $r_1 \bowtie \cdots \bowtie r_i$ (which we abbreviate as q_i) is consistent with r_{i+1} $(1 \leq i < n)$.

By an identical argument to that used to prove (6.8), except with r_i playing the role of r'_i in (6.8), it follows that $r_m = q_i[R_m]$ whenever $m \leq i$. In particular, let $m = j_{i+1}$, and let $V = R_{i+1} \cap (R_1 \cup \cdots \cup R_i)$. Since $V \subseteq R_m$, it follows that $r_m[V] = q_i[V]$. But also $r_{i+1}[V] = r_m[V]$, since r_{i+1} and r_m are consistent. Hence $r_{i+1}[V] = q_i[V]$. So r_{i+1} is consistent with q_i, which was to be shown.

(12) $\Rightarrow$ (11): This is immediate, since every monotone, sequential join expression is a monotone join expression.

(11) $\Rightarrow$ (7): Assume that $\mathbf{R}$ has a monotone join expression. We must show that every pairwise consistent database over $\mathbf{R}$ is globally consistent. Let $\mathbf{r}$ be a pairwise consistent database over $\mathbf{R}$. It is not hard to see that since no tuples are lost in joining together the relations in $\mathbf{r}$ as dictated by the monotone join expression, it follows that every member of $\mathbf{r}$ is a projection of the final result $\bowtie\mathbf{r}$. Hence $\mathbf{r}$ is globally consistent, which was to be shown.

This completes the proof. □

7. *The MVDs That Are Implied by a Join Dependency*

In this section we obtain several characterizations of sets M of MVDs (multivalued dependencies) such that M is the set of MVDs that are the consequences of a given join dependency. For simplicity of notation we shall consider only MVDs with the left-hand and right-hand sides disjoint (recall [15] that every MVD $X \twoheadrightarrow Z$ is equivalent to an MVD, namely, $X \twoheadrightarrow Z - X$, with the left-hand side and right-hand side disjoint). Thus M^+ will denote the set of all MVDs $X \twoheadrightarrow Y$, with X and Y disjoint, that are implied by the set M. We begin with some definitions.

If $\mathscr{H}$ is a hypergraph, then the set of multivalued dependencies *generated* by $\mathscr{H}$ is the set of MVDs $X \twoheadrightarrow Y$, where X and Y are (disjoint) sets of nodes and Y is the union of some connected components of the hypergraph $\mathscr{H} - X$. ($\mathscr{H} - X$ is the hypergraph obtained from $\mathscr{H}$ by deleting the set X of nodes, i.e., $\mathscr{H} - X = \{E - X: E \text{ is an edge of } \mathscr{H}\} - \{\emptyset\}$.) We then say that X *separates off* Y (from the rest of the nodes). *A* set M of multivalued dependencies is *hypergraph generated* if there is a hypergraph that generates M. Similarly, M is *graph generated* if there is a graph (treated as a hypergraph) that generates M. The following theorem is quite helpful.

THEOREM 7.1 [17, 30, 39]. *The set of MVDs implied by a join dependency* $\bowtie\mathbf{R}$ *is exactly the set of MVDs generated by the hypergraph* $\mathbf{R}$.

A multivalued dependency $X \twoheadrightarrow Y$ (with X and Y disjoint) *splits* two attributes A and B if one of them is in Y and the other is in $U - XY$, where U is the set of all the attributes. A set M of MVDs splits A and B if some MVD in M splits them.

LEMMA 7.2. *Two attributes A and B are split by a set M of MVDs if and only if they are split by its closure M^+.*

PROOF. The "only if" direction is obvious, since $M \subseteq M^+$. For the "if" direction, consider a relation r with two tuples that agree in all attributes except A and B. It is easy to see that this relation satisfies exactly those MVDs that do not split A and B. If M does not split A and B, then r satisfies M and therefore also has to satisfy M^+; hence M^+ does not split A and B. □

Thus, two logically equivalent sets of MVDs split exactly the same pairs of attributes. Given a set M of multivalued dependencies, we can construct a graph $G(M)$ with the attributes as nodes and an edge (A, B) between two attributes A and B if A and B are not split by M.

Example 7.3. Let $U = \{A, B, C, D\}$ and $M = \{A \twoheadrightarrow C, C \twoheadrightarrow D\}$. The first MVD splits C and B, and C and D, and the second MVD splits D and A, and D and B. The graph $G(M)$ of M is shown in Figure 8. □

LEMMA 7.4. *Let M be a set of MVDs, $G(M)$ its graph, and N the set of MVDs generated by $G(M)$. Then $M^+ \subseteq N$.*

PROOF. Let $X \twoheadrightarrow Y$ be an MVD in M^+. For every A in Y and B in $U - XY$, the MVD $X \twoheadrightarrow Y$ splits A and B. From Lemma 7.2, M splits A and B, and therefore

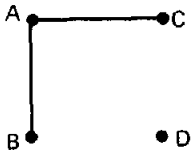


FIGURE 8

there is no edge in $G(M)$ connecting a node in Y to a node in $U - XY$. Thus X separates off Y from the rest of the nodes, and $X \twoheadrightarrow Y$ is in N. □

The converse to the lemma does not hold; in Example 7.3, the MVD $\emptyset \twoheadrightarrow D$ is generated by $G(M)$ but is not implied by M. We shall show below that the converse holds exactly for those sets of MVDs that form a cover of the set of MVDs implied by a given join dependency. (We say that M_1 is a *cover* of M_2 if $M_1^+ = M_2^+$.)

Let M be a set of MVDs. Two disjoint sets X and Y are called *orthogonal* if the MVD $U - XY \twoheadrightarrow X$ (or equivalently, by the complementation rule for MVDs [15], $U - XY \twoheadrightarrow Y$) is implied by M. It follows from Lemma 7.2 and from the rules for manipulating MVDs [5] that two attributes A and B are orthogonal (i.e., the singleton sets $\{A\}$ and $\{B\}$ are orthogonal) if and only if they are split by M. It follows from the rules for manipulating MVDs [5] that if X and Y are orthogonal, then for every pair A, B of attributes where $A \in X$ and $B \in Y$, necessarily A and B are orthogonal. We shall say that M has the *orthogonality property* if the converse also holds, that is, two sets X and Y are orthogonal whenever every attribute of X is orthogonal to every attribute of Y.

We say that M has the *intersection property* if whenever the MVDs $X \twoheadrightarrow Z$ and $Y \twoheadrightarrow Z$ are implied by M (with Z disjoint from both X and Y), then also $X \cap Y \twoheadrightarrow Z$ is implied by M.

THEOREM 7.5. *Let M be a set of multivalued dependencies. The following are equivalent:*

(1) *M is a cover of the set of MVDs implied by some join dependency.*
(2) *M^+ is hypergraph generated.*
(3) *M^+ is graph generated.*
(4) *There is exactly one graph that generates M^+.*
(5) *M^+ is the set of MVDs generated by $G(M)$.*
(6) *M has the intersection property.*
(7) *M has the orthogonality property.*

PROOF. We shall show that (1) and (2) are equivalent and (2) and (3) are equivalent. We then show that (3) ⇒ (6) ⇒ (7) ⇒ (5) ⇒ (3). Thus, conditions (1)–(3) and (5)–(7) are all equivalent. We then show that (5) ⇒ (4) ⇒ (3), which shows that (4) is equivalent to the others.

(1) ⇔ (2): If M is the set of MVDs implied by the join dependency **R**, then by Theorem 7.1 we know that M^+ is the set of MVDs generated by the hypergraph **R**.

(2) ⇒ (3): Let $\mathcal{H}$ be a hypergraph, and let $G = G(\mathcal{H})$ be the graph of $\mathcal{H}$; that is, G has the same nodes as $\mathcal{H}$ and an edge between every pair of nodes that are in the same hyperedge of $\mathcal{H}$. It is easy to see that a set X of nodes separates off another set Y in $\mathcal{H}$ iff X separates off Y in G. Therefore, the set of MVDs generated by $\mathcal{H}$ is the same as the set of MVDs generated by G.

(3) ⇒ (2): Obvious, since every graph is a hypergraph.

(3) ⇒ (6): Let G be a graph that generates M^+. Suppose that $X \twoheadrightarrow Z$ and $Y \twoheadrightarrow Z$ are in M^+. Because $X \twoheadrightarrow Z$ is in M^+, there is no edge in G connecting

a node in Z to a node in $U - XZ$. Similarly, there is no edge in G connecting a node in Z to a node in $U - YZ$. Therefore, there is no edge in G connecting a node in Z to a node in $U - [(X \cap Y) \cup Z]$. Thus $X \cap Y$ separates off Z in G, and $X \cap Y \rightarrow\!\!\!\rightarrow Z$ is in M^+.

(6) $\Rightarrow$ (7): Let $X = \{A_1, A_2, \ldots, A_k\}$ and $Y = \{B_1, B_2, \ldots, B_m\}$ be two disjoint sets with every A_i orthogonal to every B_j. Let $Z = U - XY$, let $X_i = X - A_i$ ($i = 1, \ldots, k$), and let $Y_j = Y - B_j$ ($j = 1, \ldots, m$). Since every A_i is orthogonal to every B_j, we have $ZX_iY_j \rightarrow\!\!\!\rightarrow A_i$ for each i and j. Since M has the intersection property, we have $\bigcap\{ZX_iY_j : j = 1, \ldots, m\} \rightarrow\!\!\!\rightarrow A_i$. But $\bigcap\{ZX_iY_j : j = 1, \ldots, m\} = ZX_i$. Thus $ZX_i \rightarrow\!\!\!\rightarrow A_i$ or, equivalently, by the complementation rule [15] for multivalued dependencies, $ZX_i \rightarrow\!\!\!\rightarrow Y$, for each i. Again from the intersection property, $\bigcap\{ZX_i : i = 1, \ldots, k\} \rightarrow\!\!\!\rightarrow Y$. Since $\bigcap\{ZX_i : i = 1, \ldots, k\} = Z$, we have $Z \rightarrow\!\!\!\rightarrow Y$ or, equivalently, $Z \rightarrow\!\!\!\rightarrow X$.

(7) $\Rightarrow$ (5): Suppose that M has the orthogonality property, and let N be the set of MVDs generated by $G(M)$. From Lemma 7.4, $M^+ \subseteq N$. For the other inclusion, let $X \rightarrow\!\!\!\rightarrow Y$ be an MVD in N, and let $Z = U - XY$. From the definition of N, there is no edge in $G(M)$ connecting a node in Y to a node in Z. Thus every attribute of Y is orthogonal to every attribute of Z. Therefore, because of the orthogonality property, Y is orthogonal to Z, and $X \rightarrow\!\!\!\rightarrow Y$ is in M^+.

(5) $\Rightarrow$ (3): Obvious.

(5) $\Rightarrow$ (4): Let G be a graph that generates M^+. Let A and B be attributes. A and B are split by M iff the edge (A, B) is not in G. But also, A and B are split by M iff the edge (A, B) is not in $G(M)$. Therefore $G = G(M)$.

(4) $\Rightarrow$ (3): Obvious. □

We can use Theorem 7.5 (and its proof) to give a necessary and sufficient condition for two join dependencies to imply the same set of multivalued dependencies.

COROLLARY 7.6. *Two join dependencies* $\bowtie\mathbf{R}_1$ *and* $\bowtie\mathbf{R}_2$ *imply the same set of multivalued dependencies if and only if* $G(\mathbf{R}_1) = G(\mathbf{R}_2)$.

PROOF. ($\Leftarrow$): From the proof of Theorem 7.5, the set of MVDs implied by a join dependency $\bowtie\mathbf{R}$ is equal to the set of MVDs generated by the hypergraph $\mathbf{R}$ and equal to the set of MVDs generated by the graph $G(\mathbf{R})$. The result follows easily.

($\Rightarrow$): Let M be the set of MVDs implied by $\bowtie\mathbf{R}_1$. By assumption, M is also the set of MVDs implied by $\bowtie\mathbf{R}_2$. Note that $M = M^+$, since M is clearly closed under implication. From the proof of Theorem 7.5, we see that M is generated by $G(\mathbf{R}_1)$, and, similarly, M is generated by $G(\mathbf{R}_2)$. We know that (1) of Theorem 7.5 holds. Therefore, by Theorem 7.5 we know that (4) of Theorem 7.5 holds. By (4) of Theorem 7.5 it follows that $G(\mathbf{R}_1) = G(\mathbf{R}_2)$. □

8. *Conflict-Free Sets of MVDs*

The notion of conflict-free sets of MVDs was introduced by Lien [26], who examined the relationship between the network and relational models. He showed that certain network structures can be mapped to relational structures whose semantics are described by a set of MVDs with the conflict-free property (and another additional property).

As we stated in Section 4, conflict-free sets of MVDs have several nice properties: (1) They allow a unique fourth-normal-form [15] decomposition, and (2) all MVDs

participate in the decomposition process; that is, the phenomenon where decomposing according to one MVD prevents another MVD from being applied does not occur. Furthermore, Sciore [36, 37] claims that "real-world" sets of MVDs are conflict free. He argues that if the specified set of MVDs is not conflict free, then this indicates that part of the semantics is not adequately captured, and he presents ways for enforcing conflict-freedom.

Let M be a set of multivalued dependencies. The left-hand sides of the MVDs of M are called the *keys* of M. This is, of course, a nonstandard use of the word "key." For all sets X, Y, Z, whenever the MVDs $X \twoheadrightarrow Y$ and $X \twoheadrightarrow Z$ hold (are implied by M), then also MVDs $X \twoheadrightarrow Y \cap Z$, $X \twoheadrightarrow YZ$, and $X \twoheadrightarrow Y - Z$ hold [15]; that is, the family of sets S such that $X \twoheadrightarrow S$ holds is closed under intersection, union, and set difference. A consequence of this fact is [15] that there is a partition of $U - X$ such that $X \twoheadrightarrow Y$ holds iff Y is the union of some sets in this partition. This partition is called [5] the *dependency basis* of X and is denoted by $\mathrm{DEP}(X)$.

Let us say that an MVD (or set of MVDs) *splits* a set X if it splits two attributes in X. Recall that a multivalued dependency $V \twoheadrightarrow W$ splits two attributes A and B if one of them is in W and the other is in $U - VW$, where U is the set of all the attributes. Recall also that a set M of MVDs splits A and B if some MVD in M splits them. If M is a set of MVDs, then we say that a key X of M splits attributes A and B if some MVD in M with key X splits A and B.

Definition A. A set M of MVDs is *conflict free* if

(1) M does not split its keys, and
(2) $\mathrm{DEP}(X) \cap \mathrm{DEP}(Y) \subseteq \mathrm{DEP}(X \cap Y)$; that is, those sets that are in the dependency bases of both X and Y are also in the dependency basis of $X \cap Y$.

We shall show later (Corollary 8.10) that part (2) of Definition A can be replaced by any of the equivalent conditions of Theorem 7.5. Part (2) above is a weak form of the intersection property of Section 7, restricted only to keys. We shall discuss this fact in more detail later in this section.

In Theorem 8.9 below, we shall show that a set Σ of MVDs has a conflict-free cover if and only if Σ is equivalent to an acyclic join dependency. The (weaker) fact that a conflict-free set of MVDs is equivalent to a join dependency has been shown by Sciore [37], with a different proof.

The following definition, which is not very intuitive, is technically useful.

Definition B. Let X, Y be two keys with dependency bases $\mathrm{DEP}(X)$, $\mathrm{DEP}(Y)$. X and Y are *conflict free* if

(1) $\mathrm{DEP}(X) = \{V_1, \ldots, V_k, X_1, \ldots, X_i, Z_a Y_1 \cdots Y_j\}$ and $\mathrm{DEP}(Y) = \{V_1, \ldots, V_k, Y_1, \ldots, Y_j, Z_b X_1 \cdots X_i\}$, with $Z_a X = Z_b Y$, and
(2) the sets $V_1, \ldots, V_k$ that are common to the dependency bases are also in the dependency basis of $X \cap Y$; that is, $\{V_1, \ldots, V_k\} \subseteq \mathrm{DEP}(X \cap Y)$.

Part (1) of Definition B requires that the attributes of $Y - X$ belong to the same set in the dependency basis of X, and similarly for the attributes in $X - Y$. In Lemma 8.1 below, we shall relate Definitions A and B.

We shall make use several times of the following useful inference rule [22], which Biskup [13] calls the *subset rule* for MVDs.

Subset rule for MVDs. Assume that Y and W are disjoint. Then the MVDs $X \twoheadrightarrow Y$ and $W \twoheadrightarrow V$ taken together imply the MVDs $X \twoheadrightarrow Y \cap V$ and $X \twoheadrightarrow Y - V$.

LEMMA 8.1. *The dependency bases of the keys of M have the form of part* (1) *in Definition B if and only if M does not split its keys. In particular, a set M of MVDs is conflict free by Definition A if and only if every pair of its keys is conflict free by Definition B.*

PROOF. ($\Rightarrow$): Suppose that the dependency bases of the keys of M are as in part (1) in Definition B. Let $N = \{X \twoheadrightarrow Z : X \text{ is a key of } M \text{ and } Z \in \text{DEP}(X)\}$. Then N is a cover of M, and no MVD of N splits a key. Therefore M does not split its keys.

($\Leftarrow$): Let X, Y be two keys. Let $V_1, \ldots, V_k$ be the sets common to DEP(X) and DEP(Y).

Case 1. $X - Y \neq \emptyset$ and $Y - X \neq \emptyset$. Since keys are not split, $Y - X$ is contained in one set of DEP(X), say Z, which cannot be any of the V_i's, since $V_i \cap Y = \emptyset$. Similarly, $X - Y$ is contained in a set W of DEP(Y). Let $X_1, \ldots, X_i$ be the rest of the sets in DEP(X) (other than $V_1, \ldots, V_k, Z$), and $Y_1, \ldots, Y_j$ the rest of the sets in DEP(Y) (other than $V_1, \ldots, V_k, W$). If for some m, n, we had $X_m \cap Y_n$ not equal to any of X_m, Y_n, or $\emptyset$, then X_m (respectively, Y_n) should be replaced in the dependency basis DEP(X) (respectively, DEP(Y)) by $X_m \cap Y_n$ and $X_m - Y_n$ (respectively, $X_m \cap Y_n$ and $Y_n - X_m$); this follows from the subset rule for MVDs (above) and the fact that X_m and Y_n are each disjoint from each of X and Y. Therefore, for every X_m, Y_n, either $X_m \cap Y_n = \emptyset$ or $X_m \cap Y_n = X_m = Y_n$. The last case is impossible, since the common sets are $V_1, \ldots, V_k$. Thus $X_m \cap Y_n = \emptyset$. Since $X_m \cap Y = \emptyset$ for each m and $Y_n \cap X = \emptyset$ for each n, we must have $X_m \subseteq W$ and $Y_n \subseteq Z$. Let $L = U - XYV_1 \cdots V_kX_1 \cdots X_iY_1 \cdots Y_j$. Then $Z = (Y - X)LY_1 \cdots Y_j$ and $W = (X - Y)LX_1 \cdots X_i$. Let $Z_a = (Y - X)L$ and $Z_b = (X - Y)L$. Then $Z_aX = XYL = Z_bY$.

Case 2. $X \subseteq Y$ (or $Y \subseteq X$). The arguments are similar. The idea is that DEP(Y) is a refinement of DEP(X). However, only one set of DEP(X) is refined, namely, the one that contains $Y - X$ (or $X - Y$, if $Y \subseteq X$).

Finally, the second sentence of Lemma 8.1 follows immediately from the first sentence. □

As we noted earlier, part (2) of Definition A is a weak form of the intersection property of Section 7, restricted only to keys. The definition depends on the keys of M. That is, a set of MVDs that is not conflict free may have a conflict-free cover. For example, one can always add to a conflict-free set of MVDs some redundant MVDs to destroy conflict-freedom.

Example 8.2. Suppose $M = \{E \twoheadrightarrow B, EA \twoheadrightarrow C\}$ where $U = \{A, B, C, D, E\}$. Clearly, M is conflict free: DEP(E) = $\{B, ACD\}$, and DEP(EA) = $\{B, C, D\}$. However, if we add the (redundant) MVD $EAB \twoheadrightarrow C$ to M, it is no longer conflict free, since the key EAB is split by $E \twoheadrightarrow B$. □

Lien [26] circumvents this problem by requiring that every pair of *essential keys* be conflict free. A key is *essential* if deleting from M all the MVDs that have it as the key will change the closure M^+. Lien deals with a different kind of MVDs than we do, called "MVDs with nulls," where pseudotransitivity [5] does not hold. In this case, (1) DEP(X) for a key X is determined by those MVDs whose key is contained in X, and, as a consequence, (2) two logically equivalent sets of MVDs have the same essential keys.

Since here we are dealing with ordinary MVDs, we shall first revisit some of the properties of conflict-free sets of MVDs in our context. First, let us note that

properties (1) and (2) above do not hold for sets of ordinary MVDs, even if the set is conflict free.

Example 8.3. Failure of property (1): Let $U = \{A, B, C, D\}$ and $M = \{A \twoheadrightarrow B,\ AB \twoheadrightarrow C\}$. Clearly, M is conflict free: $\mathrm{DEP}(A) = \{B, C, D\}$ and $\mathrm{DEP}(AB) = \{C, D\}$. However, the MVD $A \twoheadrightarrow C$ cannot be derived without $AB \twoheadrightarrow C$. Failure of property (2): Let $U = \{A, B, C, D, E\}$ and $N = \{E \twoheadrightarrow B,\ EAB \twoheadrightarrow C\}$. It is easy to see that N is logically equivalent to the set M of Example 8.2. Both sets are minimal covers of M^+ and N^+, and thus each key is essential for the corresponding sets. However, M and N have distinct (essential) keys, and moreover N is not even conflict-free, since the key EAB is split. □

Properties (1) and (2) above hold in some form even for conflict-free sets of ordinary MVDs. Let us call a set M of MVDs *full* if it contains all MVDs in the dependency basis of its keys; that is, if the dependency basis of a key X is $\mathrm{DEP}(X) = \{X_1, \ldots, X_n\}$, then M contains the MVDs $X \twoheadrightarrow X_i$ for $i = 1, \ldots, n$. If N is any set of MVDs, the *full version of* N is the full set M of MVDs that has the same keys as N and is logically equivalent to N. From the definition it follows that N is conflict free if and only if its full version is also conflict free.

LEMMA 8.4. *Let M be a full set of MVDs that does not split its keys. Then for every set X, the dependency basis of X is determined by the keys that are contained in X; that is, if M_X is the set of those MVDs in M whose key is contained in X, then M implies an MVD $X \twoheadrightarrow Y$ if and only if M_X implies it.*

PROOF. Let $\mathcal{F}$ be the dependency basis of X with respect to M_X. We need only show that $\mathcal{F}$ is the dependency basis of X with respect to M. Assume not; we shall derive a contradiction. Since $\mathcal{F}$ is not the dependency basis of X with respect to M, it follows from a result of Hagihara et al. [22], which is essentially a converse to the subset rule for MVDs, that there is an MVD $V \twoheadrightarrow W$ in M but not in M_X (thus $V \not\subseteq X$), and there is a set S in $\mathcal{F}$ such that $V \cap S = \emptyset$ and $W \cap S$ is neither empty nor equal to S. Since $V \cap S = \emptyset$ and $V \not\subseteq X$, there is some attribute in $V - X$ that is in a different member of $\mathcal{F}$ than S, and so there is a key $X_1 \subseteq X$ of M that splits an attribute of $V - X$ and an attribute of S. Since S is a set in $\mathcal{F}$, we know that S is not split by the key X_1; since keys are not split, $V - X_1$ is also not split by X_1. But by the subset rule for MVDs, the dependency $V \twoheadrightarrow W$ can be used to split the set containing S in the dependency basis of X_1. Thus there is a nonempty proper subset S_1 of S such that the MVD $X_1 \twoheadrightarrow S_1$ is a consequence of M. Since M is a full set of MVDs and X_1 is a key of M, it follows that the MVD $X_1 \twoheadrightarrow S_1$ is in M. By definition of M_X, it follows that this MVD is also in M_X. By augmentation [5], this MVD $X_1 \twoheadrightarrow S_1$ logically implies the MVD $X \twoheadrightarrow S_1$. Thus M_X logically implies the MVD $X \twoheadrightarrow S_1$, and so $S \notin \mathcal{F}$, since S_1 is a nonempty proper subset of S. This is a contradiction. □

LEMMA 8.5. *Let M be a full set of MVDs that does not split its keys. Then X is an essential key if and only if there are two attributes A and B that are split by an MVD implied by M with key X but not by an MVD in M with key properly contained in X.*

PROOF. ($\Rightarrow$): Let X be an essential key. Define M' to be the result of deleting from M all MVDs with key X. The dependency basis of X with respect to M must be a finer partition of $U - X$ than the dependency basis of X with respect to M', since X is an essential key. Let S' be the set in the dependency basis of X with respect to M', such that S' properly contains a set S of the dependency basis of X with respect to M. Let A be an attribute in S, and let B be an attribute in $S' - S$. Clearly, A and

B are split by the MVD $X \twoheadrightarrow S$, which is an MVD implied by M. We need only show that A and B are not split by any MVD in M with key properly contained in X. Assume not; we shall derive a contradiction. Thus we assume that A and B are split by an MVD $X' \twoheadrightarrow Y$ in M with key $X' \subsetneq X$. Since $X' \twoheadrightarrow Y$ is in M', it follows by augmentation [5] that M' implies the MVD $X \twoheadrightarrow Y$. Since $S' \cap Y$ and $S' - Y$ are each nonempty (one contains A and one contains B), this contradicts the fact that S' is in the dependency basis of X with respect to M'.

($\Leftarrow$): Let A and B be two attributes that are split by an MVD implied by M with key X but not by any MVD in M with key that is a proper subset of X. Assume that X is not an essential key; we shall derive a contradiction. By assumption, there is an MVD $X \twoheadrightarrow Y$ that is implied by M such that one of A or B is in $Y - X$ and the other is in $U - XY$. As before, define M' to be the result of deleting from M all MVDs with key X. Since X is not an essential key, and since $M \models X \twoheadrightarrow Y$, we know that $M' \models X \twoheadrightarrow Y$. By Lemma 8.4 it follows that $M'_X \models X \twoheadrightarrow Y$. Now M'_X does not split A and B, and so, by Lemma 7.2, neither does $(M'_X)^+$. However, $X \twoheadrightarrow Y$ splits A and B, and we just showed that $X \twoheadrightarrow Y$ is in $(M'_X)^+$. This is a contradiction. □

COROLLARY 8.6. *Let M and N be two logically equivalent full sets of MVDs that do not split keys. Then M and N have the same essential keys.*

PROOF. Let X be an essential key of M; we must show that X is an essential key of N. By Lemma 8.5 there is an MVD $X \twoheadrightarrow Y$ with key X in M that splits two attributes A and B that are not split by any key of M properly contained in X. Since $X \twoheadrightarrow Y$ is in M and $M^+ = N^+$, it follows that $N \models X \twoheadrightarrow Y$. So, to show that X is an essential key of N, it follows from Lemma 8.5 that we need only show that A and B are not split by any MVD $X_1 \twoheadrightarrow Y_1$ of N where X_1 is a proper subset of X. If not, then since $X_1 \twoheadrightarrow Y_1$ is in N and $M^+ = N^+$, it would follow that $M \models X_1 \twoheadrightarrow Y_1$. By Lemma 8.4 it would follow that $M_{X_1} \models X_1 \twoheadrightarrow Y_1$. But then A and B would be split by a key of M that is contained in X_1 and hence properly contained in X. This is a contradiction. □

Given a set M of MVDs, a *decomposition algorithm* replaces a relation scheme R by the two relation schemes $R \cap XY$ and $R \cap XZ$, on the basis of an MVD $X \twoheadrightarrow Y$ in M (where $Z = R - XY$); these new relation schemes can be further decomposed on the basis of another MVD in M, etc. Lien proposed [28] a decomposition algorithm which from a set M of MVDs produces a fourth normal form [15] nonredundant database scheme **R**, that is, a fourth normal form database scheme **R** that cannot be further decomposed and such that no relation scheme is contained in another relation scheme. This algorithm is a modification of Fagin's decomposition algorithm [15], where (1) the full version of the set M of MVDs is used, that is, for every key K we include all MVDs $K \twoheadrightarrow Y$, where $Y \in \mathrm{DEP}(K)$, and (2) keys are processed in nondecreasing order, that is, the keys are ordered as $K_1, \ldots, K_n$ with $K_i \not\subseteq K_j$ if $i > j$, and in the ith stage ($i = 1, \ldots, n$) the existing relation schemes are decomposed according to the dependency basis of K_i. An ordering as in (2) above is called a *p-ordering*. For example, any ordering of the keys by nondecreasing cardinality is a p-ordering. Lien [26] shows that if M is a conflict-free set of MVDs, then all p-orderings produce the same decomposition. We shall show that this decomposition has a join dependency which is equivalent to M.

LEMMA 8.7. *Let M be a conflict-free set of MVDs and $G(M)$ its graph. For any p-ordering, Lien's algorithm produces a database scheme* **R** *which consists of the maximal cliques of $G(M)$. Moreover, M is equivalent to the join dependency* $\bowtie$**R**.

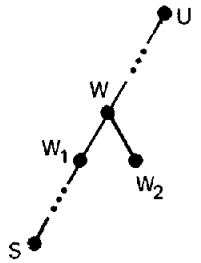


FIGURE 9

PROOF. We can consider the application of Lien's algorithm as the construction of a decomposition tree T with the nodes labeled by sets of attributes, where the root is labeled with the set U of all the attributes and the leaves are labeled with the schemes of $\mathbf{R}$. Every internal node v is labeled with a set R_v which at some stage (when v was a leaf) was decomposed by the MVDs of a key into subsets of R_v that label the sons of v. We shall identify a node of T with its label.

We begin by showing that (1) every clique of $G(M)$ is contained in some leaf of T, and (2) every leaf of T is a clique of $G(M)$.

(1) From the construction of $G(M)$, no key can split a clique of $G(M)$. The conclusion follows then, by an easy induction, from the fact that every clique of $G(M)$ is contained in the root U.

(2) Suppose that a leaf S of T contains two attributes A and B that are not adjacent in $G(M)$. Let K_i be a minimal key of M that splits A and B. Let W be the lowest (smallest) ancestor of S that contains K_i. Since K_i splits A and B (two elements of S), it follows that the ancestor of S that was a leaf at this stage (just before the MVDs with key K_i were applied) did not contain K_i. Therefore, W is an ancestor of S that was decomposed by some key K_j, with $j < i$. W is in fact a *proper* ancestor of S (i.e., $W \neq S$), since $K_i \subseteq W$ but $K_i \not\subseteq S$ (because S is a leaf). Let W_1 be the son of W in the path to S, and let W_2 be the son that contains K_i; see Figure 9. (Some son of W contains K_i, since M does not split its keys.) We have $K_i \subseteq W$, W_2 and $K_i \not\subseteq W_1$. Let $\mathrm{DEP}(K_i) = \{V_1, \ldots, V_k, X_1, \ldots, X_n, Z_a Y_1 \cdots Y_m\}$ and $\mathrm{DEP}(K_j) = \{V_1, \ldots, V_k, Y_1, \ldots, Y_m, Z_b X_1 \cdots X_n\}$ with $Z_a K_i = Z_b K_j$. Since $j < i$, it follows that $K_i - K_j \neq \emptyset$ (from the p-ordering), and so $K_i \cap Z_b \neq \emptyset$. By construction, W_2 is the intersection of W with $K_j T$, where T is a member of the dependency basis of K_j. Since W_2 contains K_i and $K_i \cap Z_b \neq \emptyset$, this member of the dependency basis is $Z_b X_1 \cdots X_n$. Therefore $W_2 = W \cap (K_j Z_b X_1 \cdots X_n)$. Thus $S \subseteq W_1 \subseteq K_j V_1 \cdots V_k Y_1 \cdots Y_m$. Since K_i splits A and B, it follows that at least one of A or B, say A, must belong to some V_t and $B \notin V_t$. (This is because from the fact that $W_1 \subseteq K_j V_1 \cdots V_k Y_1 \cdots Y_m$, it follows that the only sets of $\mathrm{DEP}(K_i)$ that can contain members of W_1 are the V_t's and one other, namely, $Z_a Y_1 \cdots Y_m$.) From conflict-freedom, M implies the MVD $K_i \cap K_j \twoheadrightarrow V_t$, and therefore, from Lemma 8.4, there is a key K_s contained in $K_i \cap K_j$ such that $K_s \twoheadrightarrow V_t$, and thus this MVD splits A and B. Since $K_i \not\subseteq K_j$, it follows that K_s is a proper subset of K_i. This contradicts the minimality of K_i.

From (1) and (2) it follows that every maximal clique of $G(M)$ is a set in $\mathbf{R}$, the database scheme produced by the algorithm. Since as Lien showed [28], the algorithm produces nonredundant database schemes, it follows from (2) that $\mathbf{R}$ is precisely the set of maximal cliques of $G(M)$. Therefore, the graph $G(\mathbf{R})$ equals $G(M)$, and the set of MVDs implied by the join dependency $\bowtie\mathbf{R}$ is equal to the set N of MVDs generated by the hypergraph $\mathbf{R}$, which is equal to the set of MVDs generated by the

graph $G(M)$. By Theorem 7.1 we know that $\bowtie\mathbf{R} \models N$, and from Lemma 7.4 we have $N \models M$. So by transitivity, $\bowtie\mathbf{R} \models M$. From the correctness of the algorithm, $M \models \bowtie\mathbf{R}$. Thus M is equivalent to the join dependency $\bowtie\mathbf{R}$. □

We shall now prove a converse to Lemma 8.7 and thereby establish the equivalence of condition (6) in Theorem 3.4 to the rest of the conditions. It is already known [17] that the set of MVDs implied by an acyclic join dependency $\bowtie\mathbf{R}$ has a cover of size polynomial in the size (number of sets) of $\mathbf{R}$. We shall construct a linear conflict-free cover.

THEOREM 8.8. *Let $\mathbf{R}$ be an acyclic hypergraph, where $\mathcal{N}$ is the set $\cup\mathbf{R}$ of nodes. The join dependency $\bowtie\mathbf{R}$ is equivalent to a conflict-free set M of multivalued dependencies with $|M| \leq min(|\mathbf{R}| - 1, |\mathcal{N}| - 1)$.*

Note. By $|S|$ we mean the number of members of set S.

PROOF. The join dependency of a hypergraph and the join dependency of its reduction are logically equivalent [7]. It follows easily that we can therefore assume without loss of generality that $\mathbf{R}$ is a reduced acyclic hypergraph.

We now show that since $\mathbf{R}$ is reduced and acyclic, it follows that $|\mathbf{R}| \leq |\mathcal{N}|$. One way of seeing this is by using Theorem 3.4(3) and Theorem 3.2. Since $\mathbf{R}$ is reduced and acyclic, it consists precisely of the maximal cliques of a chordal graph. A chordal graph has at most $|\mathcal{N}|$ maximal cliques [18], and therefore $|\mathbf{R}| \leq |\mathcal{N}|$. Another way of proving this inequality is by defining a mapping h: $\mathcal{N} \to \mathbf{R}$, where $h(A)$ is the unique set of $\mathbf{R}$ that contains A, when A is eliminated in the application of Graham's algorithm to $\mathbf{R}$. It is not hard to show then that h is onto; that is, for every R_j in $\mathbf{R}$ there is an A such that $h(A) = R_j$.

Let T be a join tree for $\mathbf{R}$. Let (R_j, R_k) be an edge of T labeled by the set $S = R_j \cap R_k$. Deletion of the edge breaks T into two subtrees T' and T''. Let $\mathcal{N}'$, $\mathcal{N}''$ be the unions of the nodes in these two subtrees, respectively. We correspond to the edge (R_j, R_k) the multivalued dependency $S \to\to \mathcal{N}' - S$. Note that the other MVD $S \to\to \mathcal{N}'' - S$ can be derived from $S \to\to \mathcal{N}' - S$. It follows easily from Theorem 7.1 that $\bowtie\mathbf{R}$ implies every such MVD, since T is a join tree and $S = \mathcal{N}' \cap \mathcal{N}''$.

Let M be the set of MVDs that correspond to the edges of T. Since T is a tree with $|\mathbf{R}|$ nodes, we have $|M| = |\mathbf{R}| - 1$ (see, e.g., [10]). We noted above that $\bowtie\mathbf{R} \models M$. It remains to show that (1) $M \models \bowtie\mathbf{R}$, and (2) M is conflict free.

(1) *M implies $\bowtie\mathbf{R}$.* The proof is by induction on $|\mathbf{R}|$. The basis ($|\mathbf{R}| = 1$) is trivial. For the induction step, suppose that the result holds for $|\mathbf{R}| \leq n - 1$, and let $\mathbf{R} = \{R_1, \ldots, R_n\}$ be an acyclic reduced hypergraph with n hyperedges. Let T be a join tree for R, and let M be the corresponding set of MVDs.

Let R_i be a leaf of T, let R_j be the node adjacent to it in the join tree, and let $S = R_i \cap R_j$. Let $\mathcal{N}'$ be the union of all nodes of T but R_i. From the definition of a join tree we have $R_i \cap \mathcal{N}' = R_i \cap R_j = S$. Let T' be the tree obtained from T by deleting R_i; T' is a join tree for $\mathbf{R}' = \mathbf{R} - \{R_i\}$, and $\mathbf{R}'$ is a reduced, acyclic hypergraph.

Let $X \to\to Y$ be the MVD in M that corresponds to an edge of T other than the edge (R_i, R_j). The MVD that corresponds to the same edge in T' is $X \to\to Y \cap \mathcal{N}'$. The MVD $X \to\to Y$ (with universe $\mathcal{N}$) implies [15] the embedded MVD $X \to\to Y \cap \mathcal{N}'$ with universe $\mathcal{N}'$; in other words, if a relation r over the set of attributes $\mathcal{N}$ satisfies $X \to\to Y$, then its projection on $\mathcal{N}'$ satisfies $X \to\to Y \cap \mathcal{N}'$.

Let r now be a relation over $\mathcal{N}$ that satisfies the set M of multivalued dependencies. The projection r' of r on $\mathcal{N}'$ satisfies the set M' of multivalued dependencies that

correspond to the edges of T'. From the induction hypothesis, r' satisfies the join dependency $\bowtie\mathbf{R}'$; that is, r' is equal to the join of its projections on the sets R_k in $\mathbf{R}$ with $k \neq i$. Since $R_k \subseteq \mathcal{N}'$, we have $r'[R_k] = r[R_k]$ for every such R_k. From the MVD of M that corresponds to the edge (R_i, R_j), we conclude that r satisfies the MVD $S \rightarrow\!\!\!\!\rightarrow R_i - S$, that is, $r = r[R_i] \bowtie r'$. Therefore, $r = r[R_1] \bowtie \cdots \bowtie r[R_n]$, and r satisfies the join dependency $\bowtie\mathbf{R}$.

(2) *M is conflict free.* Since every key of M is contained in a hyperedge of $\mathbf{R}$ (a node of T), it is easy to see that M does not split keys, and therefore M satisfies condition (1) of conflict-freedom by Lemma 8.1. Since M is equivalent to $\bowtie\mathbf{R}$, it has the intersection property (by Theorem 7.5) and therefore satisfies also condition (2) of conflict-freedom. □

The following theorem relates conflict-freedom and acyclicity.

THEOREM 8.9. *A set Σ of multivalued dependencies has a conflict-free cover if and only if Σ is equivalent to an acyclic join dependency.*

PROOF. ($\Leftarrow$): From Theorem 8.8 an acyclic join dependency is equivalent to a conflict-free set M of MVDs. Therefore, M is a conflict-free cover of Σ.

($\Rightarrow$): From Lemma 8.7, since Σ has a conflict-free cover M, it follows that Σ is equivalent to a join dependency. By Theorem 3.4, this join dependency must be acyclic. □

Note that the word "cover" is necessary in Theorem 8.9, since a set Σ of MVDs that is not conflict free may have a conflict-free cover M and therefore be equivalent to an acyclic join dependency. Also, there are cyclic join dependencies $\bowtie\mathbf{R}$ such that the set of MVDs implied by $\bowtie\mathbf{R}$ has a conflict-free cover. As a simple example, the empty set is a cover for the set of MVDs implied by the cyclic join dependency $\bowtie\{AB, BC, AC\}$.

COROLLARY 8.10. *A set M of multivalued dependencies is conflict free if and only if*

(1) *M does not split its keys, and*
(2) *M satisfies any one of the (equivalent) conditions of Theorem* 7.5.

PROOF. The first condition is the same as condition (1) of Definition A. If M is a conflict-free set of MVDs, then by Theorem 8.9 we know that M is equivalent to an (acyclic) join dependency. Thus M satisfies condition (1) of Theorem 7.5. So, if M is a conflict-free set of MVDs, then it satisfies conditions (1) and (2) of Corollary 8.10.

Conversely, assume that M satisfies conditions (1) and (2) of Corollary 8.10. Then M satisfies condition (1) of Definition A. Furthermore, M has the intersection property (this is condition (6) of Theorem 7.5). It is easy to see that the intersection property implies condition (2) of Definition A. □

ACKNOWLEDGMENTS. The authors are grateful to Marc Graham, Maria Klawe, Jeff Ullman, Moshe Vardi, and Carlo Zaniolo for helpful comments.

REFERENCES

1. AHO, A.V., BEERI, C., AND ULLMAN, J.D. The theory of joins in relational databases. *ACM Trans. Database Syst. 4*, 3 (Sept. 1979), 297–314.
2. ARMSTRONG, W.W. Dependency structures of database relationships. In *Proc. IFIP 74*, North Holland, Amsterdam, 1974, pp. 580–583

3. BACHMAN, C.W. Data structure diagrams. *Data Base 1,* 2 (1969), 4–10.
4. BATINI, C., D'ATRI, A., AND MOSCARINI, M. Formal tools for top-down and bottom-up generation of acyclic relational schemata. Proc. 7th Int. Conf. on Graph-Theoretic Concepts in Computer Science, Linz, Austria, 1981.
5. BEERI, C., FAGIN, R , AND HOWARD, J.H. A complete axiomatization for functional and multivalued dependencies in database relations. In *Proc. Int. Conf. on Management of Data* (Toronto, Ont., Can., Aug. 3–5, 1977), ACM, New York, 1977, pp. 47–61.
6. BEERI, C., FAGIN, R., MAIER, D., MENDELZON, A.O., ULLMAN, J.D., AND YANNAKAKIS, M. Properties of acyclic database schemes In *Proc. 13th Ann. ACM Symp. on Theory of Computing* (Milwaukee, Wisc , May 11–13, 1981), ACM, New York, 1981, pp 355–362.
7. BEERI, C., MENDELZON, A.O., SAGIV, Y , AND ULLMAN, J.D. Equivalence of relational database schemes. *SIAM J. Comput. 10,* 2 (June 1981), 352–370.
8 BEERI, C., AND RISSANEN, J. Faithful representation of relational database schemes Res. Rep. RJ2722, IBM Research Laboratory, San Jose, Calif., 1980.
9. BEERI, C., AND VARDI, M Y. On the properties of join dependencies. In *Advances in Database Theory,* H Gallaire, J. Minker, and J-M. Nicolas, Eds , Plenum, N.Y., 1981, pp. 25–72.
10 BERGE, C *Graphs and Hypergraphs.* North-Holland, Amsterdam, 1976.
11 BERNSTEIN, P.A , AND CHIU, D W. Using semi-joins to solve relational queries. *J ACM 28,* 1 (Jan. 1981), 25–40.
12. BERNSTEIN, P.A., AND GOODMAN, N The power of natural semijoins *SIAM J. Comput 10,* 4 (Nov. 1981), 751–771.
13 BISKUP, J. Inferences of multivalued dependencies in fixed and undetermined universe. *Theor Comput. Sci. 10* (1980), 93–105.
14. CODD, E.F. A relational model of data for large shared data banks *Commun. ACM 13,* 6 (June 1970), 377–387.
15. FAGIN, R. Multivalued dependencies and a new normal form for relational databases. *ACM Trans. Database Syst. 2,* 3 (Sept. 1977), 262–278.
16. FAGIN, R Degrees of acyclicity for hypergraphs and relational database schemes *J ACM 30,* 3 (July 1983), 514–550
17. FAGIN, R., MENDELZON, A.O , AND ULLMAN, J.D. A simplified universal relation assumption and its properties. *ACM Trans. Database Syst. 7,* 3 (Sept 1982), 343–360
18. GOLUMBIC, M.C. *Algorithmic Graph Theory and Perfect Graphs.* Academic Press, New York, 1980
19. GOODMAN, N., AND SHMUELI, O Characterizations of tree database schemas Tech. Rep., Harvard Univ., Cambridge, Mass., 1981
20. GOODMAN, N., AND SHMUELI, O. Tree queries A simple class of queries. *ACM Trans. Database Syst. 7,* 4 (Dec. 1982), 653–677.
21 GRAHAM, M H. On the universal relation Tech. Rep., Univ. of Toronto, Toronto, Ont , Can , Sept. 1979.
22. HAGIHARA, K., ITO, M., TANIGUCHI, K , AND KASAMI, T. Decision problems for multivalued dependencies in relational databases *SIAM J Comput. 8,* 2 (May 1979), 247–264.
23. HONEYMAN, P. Functional dependencies and the universal instance property in the relational model of database systems Ph.D. Dissertation, Princeton Univ., Princeton, N.J , 1980
24. HONEYMAN, P , LADNER, R.E , AND YANNAKAKIS, M Testing the universal instance assumption. *Inf. Proc. Lett. 10,* 1 (1980), 14–19
25. KORTH, H.F., AND ULLMAN, J.D. SYSTEM/U· A database system based on the universal relation assumption. Proc XP1 Workshop, Stony Brook, N.Y., June 1980.
26 LIEN, Y.E On the equivalence of database models *J. ACM 29,* 2 (Apr. 1982), 333–362
27 LIEN, Y.E. Multivalued dependencies with null values in relational data bases. In *Proc 5th Int. Conf. on Very Large Data Bases* (Rio de Janeiro, Brazil, Oct. 3–5, 1979), ACM, New York, pp. 61–66
28. LIEN, Y.E. Hierarchical schemata for relational databases. *ACM Trans. Database Syst 6,* 1 (Mar 1981), 48–69
29. MAIER, D. Discarding the universal instance assumption Preliminary results Proc XP1 Workshop, Stony Brook, N.Y., June 1980
30 MAIER, D., SAGIV, Y., AND YANNAKAKIS, M On the complexity of testing implications of functional and join dependencies. *J. ACM 28,* 4 (Oct 1981), 680–695.
31. MENDELZON, A O., AND MAIER, D. Generalized mutual dependencies and the decomposition of database relations. In *Proc. 5th Int. Conf. on Very Large Data Bases* (Rio de Janeiro, Brazil, Oct 3–5, 1979), ACM, New York, pp. 75–82
32 RISSANEN, J Theory of relations for databases—A tutorial survey. In *Proc. 7th Symp on Mathematical Foundations of Computer Science,* Lecture Notes in Computer Science 64, J. Winkowski, Ed , Springer-Verlag, pp. 537–551

33. RISSANEN, J. Independent components of relations. *ACM Trans. Database Syst. 2*, 4 (Dec. 1977), 317–325
34 RISSANEN, J. On equivalence of database schemes. In *Proc 1st ACM Conf. on Principles of Database Systems* (Los Angeles, Calif., Mar. 29–31, 1982), ACM, New York, 1982, pp. 23–26.
35. SCIORE, E The universal instance and database design. Ph D. Dissertation, Princeton, Univ, Princeton, N.J., 1980.
36. SCIORE, E. Some observations on real-world data dependencies Proc. XP1 Workshop, Stony Brook, N.Y , June 1980.
37. SCIORE, E. Real-world MVDs. In *Proc Int. Conf on Management of Data* (Ann Arbor, Mich., Apr. 29–May 1, 1981), ACM, New York, 1981, pp 121–132.
38. TARJAN, R.E., AND YANNAKAKIS, M. Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. Tech. Rep., Bell Laboratories, Murray Hill, N.J., Mar. 1982.
39. VARDI, M.Y Inferring multivalued dependencies from functional and join dependencies. Tech. Rep., Weizmann Institute, Rehovot, Israel, 1980
40. VARDI, M.Y. On decomposition of relational databases. In *Proc. 23rd IEEE Symp. on Foundations of Computer Science* (Chicago, Ill., Oct. 1982), IEEE, New York, 1982, pp 176–185.
41. VASSILIOU, Y Null values in database management—A denotational semantics approach. In *Proc. Int. Conf on Management of Data* (Boston, Mass , May 30-June 1, 1979), ACM, New York, pp 162–169.
42 WALKER, A Time and space in a lattice of universal relations with blank entries. Proc. XP1 Workshop, Stony Brook, N Y , June 1980
43. YANNAKAKIS, M. Algorithms for acyclic database schemes. In *Proc. 7th Int. Conf. on Very Large Data Bases* (Cannes, France, Sept. 9–11, 1981), ACM, New York, 1981, pp. 82–94.
44. YU, C.T , AND OZSOYOGLU, M.Z An algorithm for tree-query membership of a distributed query *Proc 1979 IEEE COMPSAC*, IEEE, N.Y., 1979, pp 306–312
45. ZANIOLO, C. Analysis and design of relational schemata for database systems. Ph.D. Dissertation, Univ of California at Los Angeles, July 1976. Available as Tech. Rep UCLA-ENG-7669.

RECEIVED MAY 1981; REVISED APRIL 1982; ACCEPTED MAY 1982