Characterizations for Acyclic Database Schemes[1]

Gösta Grahne[2]
Kari-Jouko Räihä[3]
TR 85-682
1984

Department of Computer Science
Cornell University
Ithaca, New York 14853

---

1 This work was partially supported by the Academy of Finland. Part of it was done while the outhors were at the Department of Computer Science, Univeristy of Helsinki. Apreliminary version of this paper was presented at the Third ACM SIGACT-SIGMOD Conference on Principles of Database Systems, Waterloo, Ontario, Canada, April, 1984. This version will appear in Advances in Computering Research, Vol. 3 ("The Theory of Databases", P. Kanellakis, guest ed), JAI Press, 1985.

2 INRIA, Domaine de Voluceau, Rocquencourt, Les Chesnay Cedex, France

3 Cornell University, Department of Computer Science, 405 Upson Hall, Ithaca, NY 14853. On leave from the Department of Mathematical Sciences, University of Tampere, Finland.

CHARACTERIZATIONS FOR ACYCLIC DATABASE SCHEMES[1]

by

Gösta Grahne[2]

Kari-Jouko Räihä[3]

Abstract. Acyclic database schemes have attracted much interest because of the nice properties enjoyed by such schemes. Recently some new acyclicity conditions that are strictly stronger than the normal α-acyclicity have been introduced by Fagin. Because of increased requirements, the database schemes in the new classes have some further useful properties that are not shared by α-acyclic schemes. Therefore the new classes have practical relevance.

A database designer may work in terms of attribute sets and data dependencies, and not only in terms of database schemes. Thus it is important to have a characterization for the acyclic schemes of various degree in terms of data dependencies. For α-acyclic schemes such a characterization exists, but for the new classes the question has been open. In this paper we provide characterizations for β-, γ- and Berge-acyclic database schemes. The characterizations can be stated in a simple form: thus they should be useful for the database designer.

Categories and Subject Descriptors: G.2.2 [**Discrete Mathematics**]: Graph Theory - *graph algorithms*; *trees*; H.2.1 [**Database** [redacted]**t**]: Logical Design - *schema and subschema*

General Terms: Algorithms, Design, Theory

Additional Key Words and Phrases: Acyclicity, hypergraph, database scheme, relational database, multivalued dependency, join tree

---

[1] This work was partially supported by the Academy of Finland. Part of it was done while the authors were at the [redacted] of Computer Science, University of Helsinki. A preliminary version of this paper was presented at the Third ACM SIGACT-SIGMOD Conference on Principles of Database Systems, Waterloo, Ontario, Canada, April 1984. This version will appear in *Advances in Computing Research, Vol. 3* ("The Theory of Databases", P. Kanellakis, guest ed.), JAI Press, 1985.

[2] INRIA, Domaine de Voluceau, Rocquencourt, Le Chesnay Cedex, France.

[3] Cornell University, Department of Computer Science, 405 Upson Hall, Ithaca, NY 14853. On leave from the [redacted] of Mathematical Sciences, University of Tampere, Finland.

1. Introduction

Several desirable properties, such as dependency preservation, normal forms and lossless joins have been defined for relational database schemes [BBG78]. Recently a new important concept called *acyclicity* has gained much attention [BFMY83]. It has been shown that acyclic database schemes enjoy some very useful properties. For instance, in this class of databases it is possible to join several relations in such a way that the number of tuples in the intermediate results is never decreasing. That is, we will never have the inefficient situation where an intermediate result contains a large number of tuples whereas the final result only contains a few tuples. It has even been conjectured that acyclic database schemes are the only natural ones, and that a cyclic database scheme is a sign of a design error [FMU82, Sci81].

Recently it has also been shown that there are several natural *degrees* of acyclicity for database schemes [Fag83a, Fag83b], where the previously studied concept of acyclicity (hereafter referred to as α-acyclicity) is the most general one. The strengthening of the degree of acyclicity also strengthens the related desirable properties.

Acyclic database schemes can be characterized in numerous ways. First, the definition of acyclicity can be stated in several equivalent forms. All definitions are based on the natural representation of a database scheme as a hypergraph: the attributes form the nodes of the hypergraph, and the relation schemes form the edges of the hypergraph. Some definitions of acyclicity are given using hypergraph concepts, whereas others are based on an ordinary graph constructed from a hypergraph. Various definitions for α-acyclicity can be found in e.g. [BFMY83, GoS83], and other acyclicity degrees are defined in [Fag83a, ADM85].

Many of the nice properties of acyclic database schemes can also be thought of as characterizations, in the sense that a database scheme has a particular property if and only if the scheme is acyclic. Most of the properties are related to query processing and were often discovered independently by several authors. A good sample of the properties can be found in [BFMY83, Fag83a].

A test for acyclicity gives a dynamic characterization for a class of database schemes. Because of its simplicity, the so-called Graham reduction [Gra79] is the most well-known of such tests for α-acyclicity. The method was independently presented in [YuO79]. A more efficient test is given in [TaY84]. Graham reduction is generalized for higher degrees of acyclicity in [DAM84].

All the preceding characterizations share the property that they are given in terms of the database scheme (or the hypergraph). In this paper we are mainly concerned with database design. The traditional design paradigm is based on the use of *data dependencies*, which are given as input to a design algorithm. Depending on the desired properties, one of several proposed algorithms can be used. To apply this paradigm it is useful to study the relationship between acyclicity and data dependencies. Our main results are characterizations for acyclicity in terms of data dependencies. By applying the results the designer can avoid the execution of the design algorithm before testing whether an acyclic scheme is reachable. In other words, we provide a condition that can be applied to the input of a design algorithm, so that if the dependency set satisfies the condition, the design algorithm is *guaranteed* to produce an acyclic scheme.

For α-acyclicity such a characterization already exists. If an α-acyclic database scheme has the lossless join property, then it is known that the corresponding join dependency is logically equivalent to a set of multivalued dependencies belonging to a class called *conflict-free* [BFMY83]. Furthermore, the use of a conflict-free set of multivalued dependencies in the standard decomposition process [Lie82] results in a unique database scheme that is acyclic, has the lossless join property, preserves the dependencies and in which each individual scheme is in fourth normal form. An alternative characterization, based on a subset property, is given in [GoT84].

This basic result can be generalized in different ways. In this paper we will provide similar characterizations for higher degrees of acyclicity. We will define three classes of multivalued dependencies and show, for each acyclicity degree, the equivalence between an acyclic join dependency and a set of multivalued dependencies in the corresponding class. It also follows that the decomposition process results in an acyclic database scheme of higher degree if a more restricted set of multivalued dependencies is used.

Another direction for extending the characterization based on multivalued dependencies is to consider other types of data dependencies. For instance, although each functional dependency implies a corresponding multivalued dependency, it is not always semantically satisfactory to simply replace functional dependencies by multivalued dependencies. The effect of explicitly considering functional dependencies in addition to multivalued dependencies has been studied in [Kat84, SMM84].

The rest of the paper is organized as follows. Basic definitions, notations and previous results are presented in Section 2. There we also give some simple lemmas that are useful for proving the main results. The characterizations are found in Section 3. The last section contains the conclusions.

2. Preliminaries

We assume that the reader is familiar with the basic concepts of the relational model, including the definitions of multivalued dependencies (MVDs) and join dependencies (JDs). Prior familiarity with hypergraphs would also be helpful, although not essential. In this section we will first fix some notations and then briefly review the necessary definitions and results about hypergraphs and their relation to database schemes. Further information can be found in [Ull82a, Mai83, BFMY83].

A *database scheme* for a set of attributes U (the *universe*) consists of a set $\mathbf{R} = \{R_1, \cdots, R_k\}$ of *relation schemes*, where each relation scheme R_i, $1 \le i \le k$, is a subset of U. We require that

$$\bigcup_{i=1}^{k} R_i = U \ .$$

The join dependency of the scheme is denoted by $\bowtie\mathbf{R}$ and a set of multivalued dependencies over U by M. The *keys* in M are denoted by $LHS(M)$ and they are defined as the set $\{ X \mid X \twoheadrightarrow Y \in M\}$.

The *closure* of M is denoted by M^* and defined as the set of all MVDs implied by M. A set M' is a *cover* of M if and only if $M'^* = M^*$; then we write $M' \equiv M$. A key X in $LHS(M)$ is *essential* if removing all dependencies whose key is X changes the closure of M. The *dependency basis* of a set of attributes X (with respect to a given dependency set M) is denoted by $DEP(X)$ and defined as the partition $\{Y_1, Y_2, \cdots, Y_n\}$ of $U-X$ such that if $Z \subseteq U-X$, then $X \twoheadrightarrow Z$ holds if and only if Z is the union of some of the Y_is. Finally, a set M of MVDs is *full* if for all $X \in LHS(M)$ and for all $Y \in DEP(X)$, M contains the dependency $X \twoheadrightarrow Y$.

We will now turn to defining hypergraphs and their degrees of acyclicity. The weakest form of acyclicity, α-acyclicity, has a technical definition based on several concepts of hypergraphs. We are mainly interested in the other degrees of acyclicity, but for completeness, we will give a full (albeit compact) definition of α-acyclicity.

To begin with, a *hypergraph* $\mathcal{H}$ is a pair $(\mathcal{N},\mathcal{E})$, where $\mathcal{N}$ is the finite set of *nodes* and $\mathcal{E}$ is the set of *(hyper)edges*, which are simply nonempty subsets of $\mathcal{N}$. There is a natural correspondence between hypergraphs and relation schemes: the hypergraph of $\mathbf{R}$ is the pair $(U,\mathbf{R})$, which will be denoted by $\mathbf{R}$ if no confusion arises. Similarly, the hypergraph of $LHS(M)$ is the pair $(X, LHS(M))$ (denoted simply by $LHS(M)$), where X is the set of attributes appearing in $LHS(M)$.

A *path* from edge E_1 to edge E_k is a sequence of $k \geq 1$ edges $E_1, \cdots, E_k$ such that $E_i \cap E_{i+1}$ is nonempty if $1 \leq i < k$. Two edges are *connected* if there is a path from one to the other. A set of edges is connected if every pair of edges is connected. A *connected component* is a maximal connected set of edges in $\mathcal{E}$.

The *reduction* $(\mathcal{N},\mathcal{E}')$ of $\mathcal{H} = (\mathcal{N},\mathcal{E})$ 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.

Let $\mathcal{M} \subseteq \mathcal{N}$. The edge set $\mathcal{E}_{\mathcal{M}}$ is defined as $\mathcal{E}_{\mathcal{M}} = \{ E \cap \mathcal{M} \mid E \in \mathcal{E}\} - \{\emptyset\}$. The *$\mathcal{M}$-induced hypergraph for* $\mathcal{H}$, denoted by $\mathcal{H}_{\mathcal{M}}$, is the reduction of $(\mathcal{M},\mathcal{E}_{\mathcal{M}})$. The hypergraph $\mathcal{H}-\mathcal{M}$ is defined to be the hypergraph $(\overline{\mathcal{M}},\mathcal{E}_{\overline{\mathcal{M}}})$, where $\overline{\mathcal{M}} = \mathcal{N}-\mathcal{M}$. A set $F \subseteq \mathcal{N}$ is an *articulation set* for $\mathcal{H}$ if $F = E_1 \cap E_2$ for some edges $E_1, E_2 \in \mathcal{E}$, and if $\mathcal{H}_F$ has more connected components than $\mathcal{H}$.

A *block* of a hypergraph $\mathcal{H} = (\mathcal{N},\mathcal{E})$ is an $\mathcal{M}$-induced hypergraph for $\mathcal{H}$ with no articulation set, for some $\mathcal{M} \subseteq \mathcal{N}$. A block is *trivial* if it has only one edge.

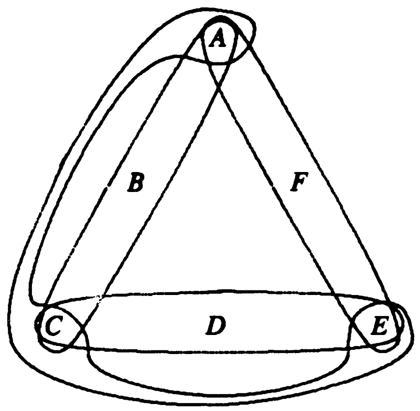

Figure 1. The hypergraph $\mathcal{H}_1$.

The usual example for illustrating the concepts defined above is hypergraph $\mathcal{H}_1$ given in Figure 1. We have $\mathcal{H}_1 = (\mathcal{N}_1,\mathcal{E}_1)$, where $\mathcal{N}_1 = \{A, B, C, D, E, F\}$ and $\mathcal{E}_1 = \{ABC, CDE, EFA, ACE\}$. (We use the standard convention of denoting e.g. the set $\{A, B, C\}$ by ABC.) The hypergraph is reduced, and its edges form a single connected component. Hypergraph $\mathcal{H}_1$ has several articulation sets: for instance, removing AC (the intersection of ABC and ACE) and taking the reduction produces the hypergraph $(BDEF, \{B, DE, EF\})$, whose edges form two connected components. Each of the edges in $\mathcal{E}_1$ gives rise to a block of $\mathcal{H}_1$. For instance, $(ACE, \{ACE\})$ is a trivial block of $\mathcal{H}_1$: it is induced by ACE and it has only one edge, thereby making the existence of an articulation set impossible.

A hypergraph $\mathcal{H}$ *generates* a set M of MVDs in the following way. The dependency $X \twoheadrightarrow Y \in M$ if and only if X and Y are disjoint sets of nodes and Y is the union of some connected components of the hypergraph $\mathcal{H}-X$. For example, removing nodes A and C from $\mathcal{H}_1$ leaves two connected components: $\{B\}$ and $\{DE, E, EF\}$. Therefore $AC \twoheadrightarrow B$, $AC \twoheadrightarrow DEF$, and $AC \twoheadrightarrow BDEF$ are in the set of MVDs generated by $\mathcal{H}_1$.

Hypergraphs can also be thought of as representations of join dependencies. Because of this close correspondence, it is understandable that the generation of MVDs was defined in such a way that the following proposition holds.

Proposition 1. [FMU82, p.353] The set of MVDs implied by join dependency $\bowtie\mathbf{R}$ is exactly the set of MVDs generated by the hypergraph $\mathbf{R}$. □

We will now turn to defining the four degrees of acyclicity. They are, in increasing strength, α-acyclicity, β-acyclicity, γ-acyclicity, and Berge-acyclicity [Fag83a]. A database scheme is acyclic if and only if the corresponding hypergraph is acyclic. We will also consider *cyclic* hypergraphs. A hypergraph is θ-cyclic if and only if it is not θ-acyclic, for $\theta = \alpha, \beta, \gamma$, or Berge. Finally, we say that a join dependency $\bowtie\mathbf{R}$ is θ-acyclic precisely when the hypergraph $\mathbf{R}$ is θ-acyclic.

The weakest form of acyclicity is α-acyclicity, defined as follows.

Definition 1. [BFMY83, p.483] A reduced hypergraph is α-*acyclic* if and only if all its blocks are trivial. A hypergraph is α-acyclic precisely when its reduction is α-acyclic. □

It is easy to check that hypergraph $\mathcal{H}_1$ given in Figure 1 is α-acyclic. For instance, we already saw that the *ABCDEF*-induced hypergraph for $\mathcal{H}_1$, i.e. $\mathcal{H}_1$ itself, does have an articulation set. Therefore $\mathcal{H}_1$ is not a block. By trying all possible inducers for producing blocks it can be seen that all blocks are trivial, i.e. have only one edge.

Note that the subhypergraph where edge *ACE* is omitted does not have an articulation set, and it has three edges. However, that hypergraph is not induced by any node set $\mathcal{M}$, and therefore it is not a block. On the other hand, it shows the somewhat counterintuitive property that an α-acyclic hypergraph can have an α-cyclic subhypergraph. In other words, an α-cyclic hypergraph can be made α-acyclic by adding new edges!

The key concept used in the proofs of our characterizations is that of a join tree, which we define next.

Definition 2. [BFMY83, p.486] A *join tree* for a database scheme **R** is a tree with set **R** of nodes, such that

(i) each edge (R_i, R_j) is labelled by the set of attributes $R_i \cap R_j$, and

(ii) for every pair R_i, R_j $(R_i \neq R_j)$ and for every $A \in R_i \cap R_j$, each label of an edge along the unique path between R_i and R_j contains A. This path is called an A-labelled path. □

Note that not every database scheme has a join tree, and that there might be several join trees for one database scheme. Figure 2 shows one possible join tree for the database scheme given in Figure 1.

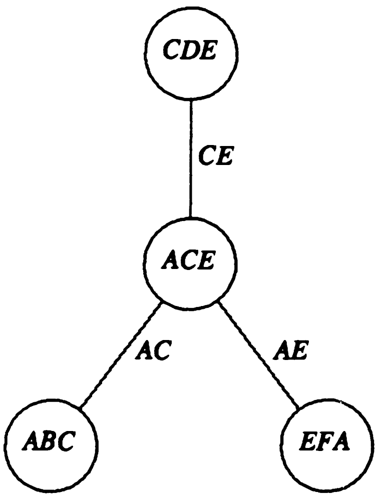

Figure 2. A join tree for $\mathbf{R} = \{ABC, CDE, EFA, ACE\}$.

A join tree *corresponds* to a set of multivalued dependencies in the following way.

Definition 3. [BFMY83, p.510] An MVD corresponding to an edge (R_i, R_j) of a join tree T for a database scheme **R** is $R_i \cap R_j \twoheadrightarrow \mathcal{N} - R_i \cap R_j$, where $\mathcal{N}$ is the union of the nodes in one of the two subtrees that are obtained by deleting the edge (R_i, R_j) from T. The set of MVDs corresponding to a join tree is the union of the MVDs corresponding to the edges of the tree. □

For instance, a set of MVDs corresponding to the join tree in Figure 2 is $\{AE \twoheadrightarrow F, AC \twoheadrightarrow B, CE \twoheadrightarrow D\}$. Note that the definition does not determine whether e.g. the dependency $AE \twoheadrightarrow F$ or its complement, $AE \twoheadrightarrow BCD$, corresponds to the edge AE. Only one of them is included in any set of MVDs corresponding to the edges of the tree, since the other can always be derived using the axioms for MVDs.

We next define some notions describing the effect of a dependency set on attribute sets.

Definition 4. Let M be a set of MVDs.

(a) A dependency $X \twoheadrightarrow Y$ in M *splits* a set V if $V \cap Y \neq \varnothing$ and $(U-XY) \cap V \neq \varnothing$. A set X in $LHS(M)$ splits V if some $X \twoheadrightarrow Y$ in M splits V. The set M splits V precisely when some $X \twoheadrightarrow Y$ in M splits V.

(b) A dependency $X \twoheadrightarrow Y$ in M *separates* attributes A and B if $X \twoheadrightarrow Y$ splits AB. A set X in $LHS(M)$ separates A and B if some $X \twoheadrightarrow Y$ in M separates A and B. A set X in $LHS(M)$ separates attribute sets P and Q if X splits AB for all $A \in P$ and $B \in Q$. □

As an example, let $U = ABCDE$ and consider the dependency $AB \twoheadrightarrow CD$. The dependency splits the sets DE and BDE, but it does not split either BD or BE. It separates e.g. C and E.

Proposition 2. [BFMY83, p.508] Let M and N be two equivalent full sets of MVDs that do not split keys. Then M and N have the same essential keys. □

Definition 5. [BFMY83, p.505] A set M of MVDs is *conflict-free* if

(i) no key in $LHS(M)$ is split by M, and

(ii) for all keys X and Y in $LHS(M)$, $DEP(X) \cap DEP(Y) \subseteq DEP(X \cap Y)$. □

For example, the set $\{AE \twoheadrightarrow F, AC \twoheadrightarrow B, CE \twoheadrightarrow D\}$ is conflict-free since the keys are not split, and since condition (ii) of the definition is trivially satisfied because the dependency bases for the keys have no common elements.

Our next two results from [BFMY83] relate the concepts defined above.

Proposition 3. [BFMY83, p.492] A hypergraph **R** has a join tree if and only if **R** is α-acyclic. □

Proposition 4. [BFMY83, p.510] The set of MVDs corresponding to a join tree for an α-acyclic hypergraph **R** is conflict-free and equivalent to $\bowtie \mathbf{R}$. □

In the example that we have elaborated we saw that the database scheme $\{ABC, CDE, EFA, ACE\}$ had an α-acyclic hypergraph $\mathcal{H}_1$ (Figure 1) and a join tree (Figure 2). It is easy to verify that $\bowtie\{ABC, CDE, EFA, ACE\}$ and $\{AE \twoheadrightarrow F, AC \twoheadrightarrow B, CE \twoheadrightarrow D\}$ are equivalent. Proposition 1 can be used to check that the JD implies each of the MVDs, and the converse implication can be verified by using the *chase* introduced in [ABU79].

MVDs can be used to decompose a relation scheme losslessly into subschemes [Fag77]. If the set of MVDs is conflict-free, there is a particularly attractive decomposition algorithm proposed by Lien [Lie81, Lie82]. It is repeated below.

The correctness and several other properties of Algorithm 1 are proved in [Lie82]. The resulting decomposition is nonredundant (i.e. no relation scheme is contained in another relation

---

Algorithm 1. [Lie82] Decomposition of a database scheme $\{U\}$.

Input: A conflict-free set of MVDs M and a p-ordering q: $X_1, \ldots, X_k$ which includes all the keys in $LHS(M)$. (The sequence q is a p-ordering if $X_i \subsetneq X_j$ implies $1 \le i \le j \le k$.)

Output: A collection **R** of relation schemes such that the decomposition is lossless and the schemes are in fourth normal form.

Method:

```
1. {Initialize}
   R := {U};
2. {Decompose using each set in q}
   for i := 1 to k do begin
       Δ := ∅;
       for each Z ∈ R such that X_i ⊂ Z do begin
           R := R−Z;
           Δ := Δ ∪ {X_i(Z∩W) | Z∩W ≠ ∅ for some W ∈ DEP(X_i)};
       end;
       R := R ∪ Δ;
   end;
```

---

scheme) and unique. For acyclicity, the following result is of particular interest.

Proposition 5. [BFMY83, p.508] For any conflict-free set M, Algorithm 1 produces a database scheme $\mathbf{R}$ such that $M = \bowtie\mathbf{R}$. □

Corollary 1. [BFMY83, p.511] A set M of MVDs has a conflict-free cover if and only if M is equivalent to an α-acyclic join dependency. □

The following property of Algorithm 1 is also useful for our proofs.

Lemma 1. Let M be the set of MVDs corresponding to an arbitrary (but fixed) join tree for a database scheme $\mathbf{R}$ generated by Algorithm 1, and let X be an arbitrary key in $LHS(M)$. Then X is essential. Moreover, if R_i and R_j belong to different components in the join tree after the X-labelled edge is removed, and if $R_i \cap R_j \neq \emptyset$, then X separates $R_i - R_j$ and $R_j - R_i$.

Proof. Let the relation schemes connected by the X-edge be R and R'. Because $\mathbf{R}$ is nonredundant, $R \not\subseteq R'$ and $R' \not\subseteq R$. Let $A \in R - R'$ and $A' \in R' - R$. By the definition of join trees, $A \notin X$ and $A' \notin X$. Let $X \twoheadrightarrow \mathcal{N} - X$ be the MVD in M that corresponds to X, where $\mathcal{N}$ is the set of attributes in one of the components obtained by deleting the X-edge from the join tree. Without loss of generality, assume that $A \in \mathcal{N} - X$. It follows that $A' \notin \mathcal{N} - X$: otherwise the definition of join trees would require that $A' \in X$, which is not true. Thus X splits AA'.

On the other hand, removing any other edge than X from the join tree would leave R and R' (and thus A and A') in the same component. Therefore X is the *only* key in $LHS(M)$ that splits AA'. Removing from M the dependency that corresponds to X would thus change the closure of M, implying that X is essential.

The proof that X separates $R_i - R_j$ and $R_j - R_i$ is similar to the proof that X splits AA'. □

The rest of this section will be used to define stronger degrees of acyclicity. We begin with β-acyclicity.

Definition 6. [Fag83, p.521] A *weak* β*-cycle* in a hypergraph $\mathbf{R}$ is a sequence $(S_1, x_1, \cdots, S_m, x_m, S_{m+1})$ such that

(i) $x_1, \cdots, x_m$ are distinct nodes of $\mathbf{R}$;

(ii) $S_1, \cdots, S_m$ are distinct edges of $\mathbf{R}$, and $S_{m+1} = S_1$;

(iii) $m \geq 3$, that is, there are at least three edges involved; and

(iv) x_i is in S_i and S_{i+1} $(1 \leq i \leq m)$, and in no other S_j.

A hypergraph is β*-acyclic* if and only if it has no weak β-cycle. □

The α-acyclic hypergraph of Figure 1 is β-cyclic since the sequence $(ABC, C, CDE, E, EFA, A, ABC)$ is a weak β-cycle. We see that C is only in ABC and CDE, E is only in CDE and EFA, and A is only in EFA and ABC.

There exist several equivalent definitions for β-acyclicity. (This explains why the modifier "weak" is used in the definition: the related concept of a β-cycle is defined in [Fag83a].) We have chosen the definition above because it is most convenient for our proofs, and because it best illustrates the small differences in the definitions of the various acyclicity degrees. Our next result shows that the definition can be slightly weakened: the distinctness of the S_i-edges is implied by the other conditions.

Lemma 2. Let $\mathbf{R}$ be a hypergraph and $(S_1, x_1, \cdots, S_m, x_m, S_{m+1})$ a sequence that satisfies conditions (i) through (iv) of Definition 6, except possibly the distinctness of the S_i sets. Then the sequence is a weak β-cycle in $\mathbf{R}$.

Proof. Let i and k be arbitrary such that $1 \leq i < k \leq m$; we must show that $S_i \neq S_k$. By condition (iv), x_i is in S_i and S_{i+1}, and in no other S_j. Thus S_i can equal S_k only if $k = i+1$. But then condition (iv) again implies that x_k is in S_k and S_{k+1}, and in no other S_j. Because of condition (iii) S_{k+1} cannot be S_i even if $k+1 = m$. Therefore x_k is not in S_i, yielding the result. □

For γ-acyclicity we make the following definition.

Definition 7. [Fag83a, p.524] A *γ-cycle* in a hypergraph **R** is a sequence $(S_1, x_1, \cdots, S_m, x_m, S_{m+1})$ such that

(i) $x_1, \cdots, x_m$ are distinct nodes of **R**;

(ii) $S_1, \cdots, S_m$ are distinct edges of **R**, and $S_{m+1} = S_1$;

(iii) $m \geq 3$;

(iv) x_i is in S_i and S_{i+1} $(1 \leq i \leq m)$; and

(v) if $1 \leq i < m$, then x_i is in no S_j except S_i and S_{i+1}.

A hypergraph is *γ-acyclic* if and only if it has no γ-cycle. □

Figure 3 shows a γ-cyclic hypergraph. Note that this hypergraph is β-acyclic. The γ-cycle in the hypergraph is given by the sequence $(ABC, A, ACE, E, CDE, C, ABC)$. Now A is only in ABC and ACE and E is only in ACE and CDE. Since C is in all three edges the γ-cycle is not a weak β-cycle.

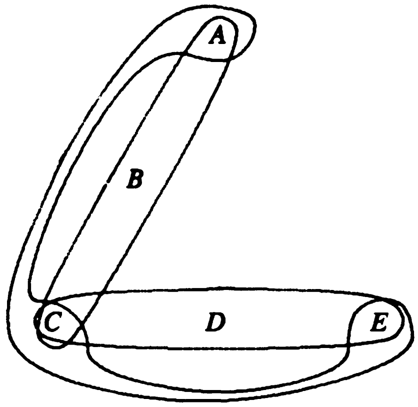

Figure 3. A γ-cyclic hypergraph.

Finally we give the definition for Berge-acyclicity.

Definition 8. [Fag83a, p.519] A *Berge-cycle* in a hypergraph **R** is a sequence $(S_1, x_1, \cdots, S_m, x_m, S_{m+1})$ such that

(i) $x_1, \cdots, x_m$ are distinct nodes of $\mathbf{R}$;

(ii) $S_1, \cdots, S_m$ are distinct edges of $\mathbf{R}$, 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-acyclic* if and only if it has no Berge-cycle. □

An example of a Berge-cyclic (but γ-acyclic) hypergraph is shown in Figure 4.

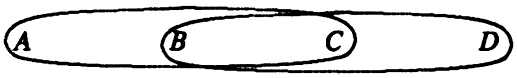

Figure 4. A Berge-cyclic hypergraph.

The Berge-cycle consists of the sequence (ABC, B, BCD, C, ABC). The sequence is not a γ-cycle since only two edges are involved. For this example the Berge-cyclicity is also immediately implied by the following result.

Proposition 6. [Fag83a, p.519] If some pair of edges of a hypergraph has two or more nodes in common, the hypergraph is Berge-cyclic. □

We have seen in the examples that there are hypergraphs that are α-acyclic but β-cyclic, β-acyclic but γ-cyclic, and γ-acyclic but Berge-cyclic. Going from α-acyclicity to β-, γ- and Berge-acyclicity properly strengthens the requirements. This can also be seen from the definitions. The following result holds.

Proposition 7. [Fag83a, p.527] Berge-acyclicity $\Rightarrow$ γ-acyclicity $\Rightarrow$ β-acyclicity $\Rightarrow$ α-acyclicity. None of the reverse implications holds. □

3. Characterizations

We will illustrate the characterization for each degree of acyclicity by an example in connection with the corresponding theorem. If the reader wants to skip the proofs, he is advised to simply read the results and the surrounding discussion.

We start with a technical lemma that is useful for proving the characterization for β-acyclicity.

Lemma 3. Let **R** be an α-acyclic hypergraph and M the set of MVDs corresponding to a join tree for **R**. If the hypergraph *LHS*(M) has a weak β-cycle $(S_1, x_1, \cdots, S_m, x_m, S_{m+1})$, then none of the paths in the join tree can contain three edges whose distinct labels belong to $\{S_1, \cdots, S_m\}$.

Proof. Recall that by the construction of M, each element of *LHS*(M) (in particular, each S_i) is the label of an edge in the join tree. Suppose to the contrary that an edge labelled with S_k belongs to the path that connects the edges labelled with S_i and S_j (see Figure 5; the ancestor relation of the nodes is irrelevant for our purposes).

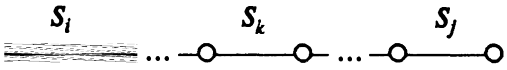

Figure 5. A path in the join tree.

We will use induction on $|j-i|$ to derive a contradiction.

Basis. $|j-i| = 1$. Without loss of generality, suppose that $j = i+1$. By the definition of a weak β-cycle, x_i is in S_i and S_{i+1} and in no other S_k. The definition of a join tree implies that every edge on the path from the S_i-edge to the S_j-edge (= S_{i+1}-edge) is labelled with x_i. In particular, $x_i \in S_k$: a contradiction.

Induction step. Suppose that the claim holds for any S_h and $S_{h'}$ with $1 \le |h'-h| \le p$; we will show that it holds for S_i and $S_j = S_{i+p+1}$.

By the induction hypothesis, the path from the S_{i+p}-edge to the S_j-edge cannot contain the S_k-edge (since $j-(i+p) = 1 \leq p$). This leaves the possibilities shown in Figure 6.

In all cases, the S_k-edge belongs to the path from the S_i-edge to the S_{i+p}-edge. This contradicts the induction hypothesis, since $(i+p)-i = p$. □

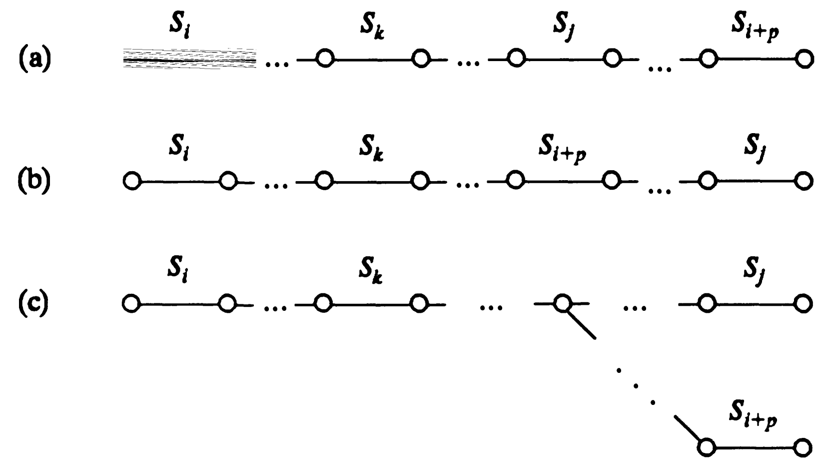

Figure 6. Refinements of the path in Figure 5.

We are now ready to attack the first degree of acyclicity, β-acyclicity. What conditions should the equivalent set of MVDs satisfy? Clearly, it must have a conflict-free cover (recall that by Corollary 1 this was the characterization for α-acyclicity). What are the additional properties required? To get some intuitive feeling of the problem, consider the β-cyclic join dependency $\bowtie\{ABC, CDE, EFA, ACE\}$ (see Figure 1). An equivalent set of MVDs is $\{AE \twoheadrightarrow F, AC \twoheadrightarrow B, CE \twoheadrightarrow D\}$. A closer look at the keys of this set reveals a weak β-cycle given by the sequence $(AE, A, AC, C, CE, E, AE)$. It turns out that this cycle among the keys is the source of the β-cyclicity of the JD. We have the following general result.

Theorem 1. A set of multivalued dependencies M' is equivalent to a β-acyclic join dependency $\bowtie\mathbf{R}$ if and only if M' has a cover M with the following properties:

(a) M is conflict-free, and

(b) $LHS(M)$ is β-acyclic.

Proof. *Only if.* Suppose that $M' = \bowtie\mathbf{R}$ where $\mathbf{R}$ is β-acyclic; we must show that M' has a cover M satisfying (a) and (b). By Proposition 7, $\mathbf{R}$ is also α-acyclic, and thus $\mathbf{R}$ has a join tree (by Proposition 3). Let M be the set of MVDs that corresponds to a join tree for $\mathbf{R}$. By Proposition 4, M is conflict-free and $M = \bowtie\mathbf{R}$. Therefore M is a cover of M', and it remains to show that (b) holds for M.

Suppose not. Then there exists a weak β-cycle $(S_1, x_1, \cdots, S_m, x_m, S_{m+1})$ of $LHS(M)$ such that

(i) $x_1, \cdots, x_m$ are distinct nodes of the hypergraph $LHS(M)$;

(ii) $S_1, \cdots, S_m$ are distinct edges of $LHS(M)$, and $S_{m+1} = S_1$;

(iii) $m \geq 3$; and

(iv) x_i is in S_i and S_{i+1} $(1 \leq i \leq m)$, and in no other S_j.

By the construction of M, each S_i $(1 \leq i \leq m)$ is the label of an edge (R_i, R_i') of the join tree so that $S_i = R_i \cap R_i'$. The join tree must have a unique path that connects any two distinct edges. Let $(R_{i,0}, R_{i,1}, \cdots, R_{i,n_i}, R_{i,n_i+1})$ be the path that connects the edges labelled with S_i and S_{i+1}; here $n_i \geq 1$, $R_{i,0} \cap R_{i,1} = S_i$, $R_{i,n_i} \cap R_{i,n_i+1} = S_{i+1}$, and $(R_{i,j}, R_{i,j+1})$ is an edge of the join tree for $0 \leq j \leq n_i$. Define $R_{m+1,0} = R_{1,0}$.

We claim that $(R_{1,0}, x_1, \cdots, R_{m,0}, x_m, R_{m+1,0})$ is a weak β-cycle of $\mathbf{R}$ (the x_i:s are the same as in the weak β-cycle of $LHS(M)$). Since we assumed that $\mathbf{R}$ is β-acyclic, this yields a contradiction and proves the only-if part of the claim. We will deal with the four conditions one at a time.

(i) We must show that $x_1, \cdots, x_m$ are distinct nodes of $\mathbf{R}$. This follows immediately from the facts that $x_1, \cdots, x_m$ are distinct nodes of $LHS(M)$ and $\cup\, LHS(M) \subseteq \cup\mathbf{R}$.

(iii) $m \geq 3$. Obvious by the construction.

(iv) We claim that x_i is in $R_{i,0}$ and $R_{i+1,0}$ $(1 \le i \le m)$, and in no other $R_{j,0}$. To see that $x_i \in R_{i,0} \cap R_{i+1,0}$, simply recall that

$$x_i \in S_i \cap S_{i+1} = (R_{i,0} \cap R_{i,1}) \cap (R_{i+1,0} \cap R_{i+1,1}) \subseteq R_{i,0} \cap R_{i+1,0} \, .$$

Suppose then that $x_i \in R_{j,0}$ for some $j \neq i, i+1$. Because $x_i \in R_{i,0}$, property (ii) of Definition 2 implies that x_i belongs to the label of every edge on the path from $R_{i,0}$ to $R_{j,0}$. Since $x_i \notin S_j$, the edge labelled with S_j does not belong on this path. Thus we have one of the possibilities shown in Figure 7.

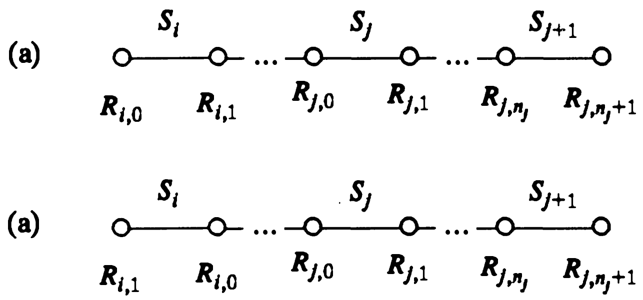

Figure 7. Two possible paths from $R_{i,0}$ to R_{j,n_j+1}.

In either case, we see that the edge labelled with S_j belongs to the path that connects edges S_i and S_{j+1}, contradicting Lemma 3. Therefore x_i cannot be in $R_{j,0}$.

(ii) Since $R_{1,0}, \cdots, R_{m,0}$ are nodes of the join tree, they are edges of $\mathbf{R}$. Their distinctness follows from Lemma 2.

If. Suppose that $M' \equiv M$, where M satisfies (a) and (b). We must show that there exists a β-acyclic join dependency $\bowtie\mathbf{R}$ such that $M' \equiv \bowtie\mathbf{R}$.

By property (a), M is conflict-free. Clearly we can (without violating the conflict-freedom property) remove from M dependencies for nonessential keys and pad the resulting set with additional right-hand sides to obtain a full set of dependencies M'' such that all the keys in $LHS(M'')$ are essential. Then we can apply Algorithm 1 to M'' to produce a database scheme $\mathbf{R}$. By Proposition 5, $\mathbf{R}$ is α-acyclic and $M'' \equiv \bowtie\mathbf{R}$, and thus also $M' \equiv M \equiv M'' \equiv \bowtie\mathbf{R}$. If $\mathbf{R}$ is β-acyclic, we are done. Suppose therefore that $\mathbf{R}$ is β-cyclic, i.e. there exists a weak β-cycle $(R_1, x_1, \cdots, R_m, x_m, R_{m+1})$ in $\mathbf{R}$. Because $\mathbf{R}$ is α-acyclic, it has a join tree (by Proposition 3). For each $1 \leq i \leq m$, let S_i be the label of the first edge on the unique path from R_i to R_{i+1} in the join tree, and define $S_{m+1} = S_1$. We will show that $(S_1, x_1, \cdots, S_m, x_m, S_{m+1})$ is a weak β-cycle of $LHS(M'')$, and thus of $LHS(M)$. This contradicts property (b) of M and proves the claim.

(i) We must show that $x_1, \cdots, x_m$ are distinct nodes of the hypergraph $LHS(M'')$. They are clearly distinct; we must only show that for $1 \leq i \leq m$, $x_i \in X$ for some $X \in LHS(M'')$.

We know that $x_i \in R_i \cap R_{i+1}$ for $1 \leq i \leq m$. By the definition of a join tree, x_i belongs to the label of every edge on the path from R_i to R_{i+1} in the join tree. In particular, $x_i \in S_i$ (the label of the first edge on the path).

Let M^T be the set of MVDs corresponding to the join tree. By Lemma 1, all the keys in $LHS(M^T)$ are essential. By Proposition 4, M^T is conflict-free and $M^T \equiv \bowtie\mathbf{R}$. Extending M^T to a full set of MVDs changes none of these properties. Therefore, by Proposition 2, $LHS(M^T) = LHS(M'')$, proving that $S_i \in LHS(M'')$ for $1 \leq i \leq m$.

(ii) $S_1, \cdots, S_m$ must be distinct edges of $LHS(M'')$. We already showed that they belong to $LHS(M'')$. Their distinctness follows again from Lemma 2 (after we have shown properties (iii) and (iv) below).

(iii) $m \geq 3$. Obvious by the construction.

(iv) Finally, we must show that x_i is in S_i and S_{i+1}, and in no other S_j. We already saw that $x_i \in S_i$. Like in the proof of condition (iv) for the only-if part, we see that S_{i+1} must be the label of the last edge on the path from R_i to R_{i+1}. Otherwise the path from R_i to R_{i+2} (or R_2 if $i=m$) would contain the edges labelled with S_i, S_{i+1}, and S_{i+2}, contradicting Lemma 3. Therefore we have also $x_i \in S_{i+1}$.

Because $x_i \notin R_j$ when $j \neq i, i+1$ (by the definition of the weak β-cycle in $\mathbf{R}$), we also have $x_i \notin R_j \cap T$ for any $T \in \mathbf{R}$ when $j \neq i, i+1$. In particular, $x_i \notin S_j$ when $j \neq i, i+1$. □

The characterization for β-acyclicity leaves something to be desired, since β-acyclicity of $\bowtie\mathbf{R}$ is characterized in terms of β-acyclicity of $LHS(M)$. Note, however, that $LHS(M)$ is defined in terms of the dependency set, not in terms of the database scheme: therefore the characterization satisfies the basic requirement that we made. Moreover, $LHS(M)$ is likely to be a smaller hypergraph than $\mathbf{R}$, thus making the test easier and more efficient.

Fortunately, for other degrees of acyclicity we are able to give direct characterizations that are not based on the acyclicity of some other hypergraph. γ-cycles in join dependencies are not directly generated by cycles in the keys of the set of equivalent MVDs. The keys, however, play an important role in this context, too, as the following theorem shows.

Theorem 2. A set of multivalued dependencies M' is equivalent to a γ-acyclic join dependency $\bowtie\mathbf{R}$ if and only if M' has a cover M with the following properties:

(a) M is conflict-free, and

(b) for any two distinct keys X and Y in $LHS(M)$ whose intersection is non-empty, $X \cap Y \in LHS(M)$ and $X \cap Y$ separates $X-Y$ and $Y-X$.

To illustrate the intuitive idea behind the conditions we will give an example before proving the theorem. The JD equivalent to

$$\{ AB \twoheadrightarrow C, AB \twoheadrightarrow D, AE \twoheadrightarrow F, AE \twoheadrightarrow G \}$$

is shown as a hypergraph in Figure 8(a). The set of MVDs does not satisfy condition (b) of the theorem, since the intersection of AB and AE is non-empty and no key A belongs to the set. The hypergraph of Figure 8(a) is γ-cyclic. By adding the dependency $A \twoheadrightarrow BCD$ to the set of MVDs we get a set that satisfies the conditions of Theorem 2. Note in particular that $A \twoheadrightarrow BCD$ separates B and E. The equivalent JD is now $\bowtie\{ABC, ABD, AEF, AEG\}$, which is γ-acyclic. The hypergraph is shown in Figure 8(b). One can note that adding the dependency $A \twoheadrightarrow BCD$ to the set of MVDs will make the set ABE unnecessary in the equivalent JD. This is because the new dependency can be used to decompose ABE into AB and AE, which disappear as subsets of other schemes. Thus the γ-cycle will be removed.

(a)

(b)

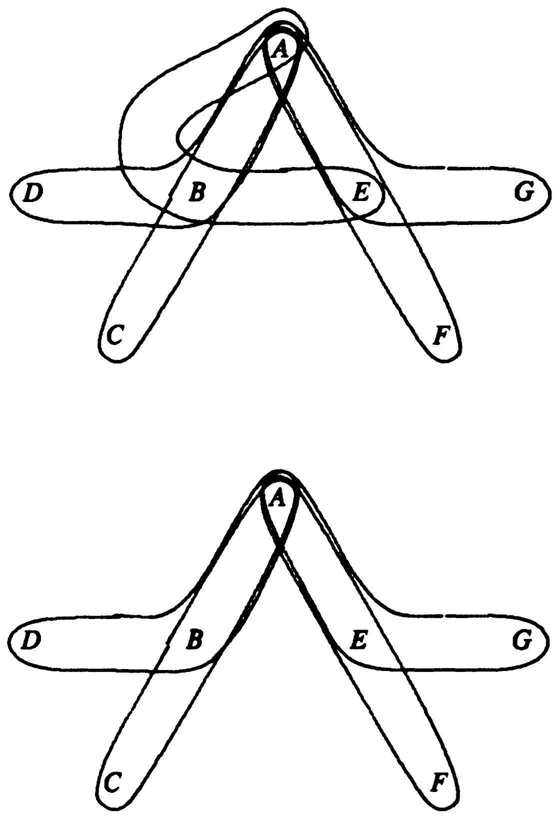

Figure 8. Hypergraphs for $\{ABC, ABD, AEF, AEG, AEB\}$ and $\{ABC, ABD, AEF, AEG\}$.

Proof of Theorem 2. *Only if.* Suppose that $M' = \bowtie\mathbf{R}$ where $\mathbf{R}$ is γ-acyclic; we must show that M' has a cover M satisfying (a) and (b). Like in the proof of Theorem 1, let M be the set of MVDs that corresponds to a join tree for $\mathbf{R}$. Then (a) is immediate.

To see that (b) holds, we will first show that $X \cap Y \in LHS(M)$ whenever $X \cap Y \neq \emptyset$ for distinct X and Y in $LHS(M)$. Suppose to the contrary that $\emptyset \neq X \cap Y \notin LHS(M)$ for some X and Y in $LHS(M)$. In particular, $X \cap Y \neq X$ and $X \cap Y \neq Y$; therefore $X - Y \neq \emptyset$ and $Y - X \neq \emptyset$.

Consider the join tree for $\mathbf{R}$, and the path $(R_0, R_1, \cdots, R_k)$ that connects the edges labelled with X and Y; i.e. $R_0 \cap R_1 = X$ and $R_{k-1} \cap R_k = Y$. By Definition 2, the path from R_0 to R_k is an $X \cap Y$-labelled path. Let (R_{i-1}, R_i) be the last edge (going from R_0 to R_k) whose label contains an attribute of $X-Y$. Such an edge (with $i<k$) must belong to the path, since $R_{k-1}\cap R_k = Y$ does not contain attributes of $X-Y$. Let A be an attribute of $X\cap Y$ and B an attribute of $(X-Y)\cap(R_{i-1}\cap R_i)$.

Consider then the edge (R_i, R_{i+1}). We know that $X\cap Y \subseteq R_i\cap R_{i+1}$. The containment must be proper: otherwise $X\cap Y$ would be the label of an edge in the join tree, contrary to the assumption that $X\cap Y \notin LHS(M)$. Therefore $(R_i\cap R_{i+1}) - (X\cap Y) \neq \varnothing$; let C be an arbitrary attribute of $(R_i\cap R_{i+1}) - (X\cap Y)$. By the definition of i, $(R_i\cap R_{i+1})\cap(X-Y) = \varnothing$. Thus $C \notin X$, which combined with property (b) of join trees (Definition 2) implies that $C \notin R_0$.

Combining the properties of the attributes defined above, we see that $A \in R_0$, $A \in R_i$, and $A \in R_{i+1}$; $B \in R_0$, $B \in R_i$, and $B \notin R_{i+1}$; and $C \notin R_0$, $C \in R_i$, and $C \in R_{i+1}$. It is now straightforward to verify that $(R_0, B, R_i, C, R_{i+1}, A, R_0)$ satisfies all the conditions for a γ-cycle. Thus $\mathbf{R}$ is γ-cyclic; a contradiction. Therefore $X\cap Y \in LHS(M)$.

The fact that $X\cap Y$ separates $X-Y$ and $Y-X$ follows from Lemma 1.

If. Suppose that $M' \equiv M$, where M satisfies (a) and (b). We must show that there exists a γ-acyclic join dependency $\bowtie\mathbf{R}$ such that $M' \equiv \bowtie\mathbf{R}$.

We begin by showing that (a) and (b) imply that $LHS(M)$ is β-acyclic. Suppose not; then $LHS(M)$ has a weak β-cycle $(S_1, x_1, \cdots, S_m, x_m, S_{m+1})$. By the definition, $\{x_i, x_{i+1}\} \subseteq S_{i+1}$ for $1\le i\le m$ (where we define $x_{m+1} = x_1$). Consider an arbitrary S_j with $1\le j\le m$. Because $S_j\cap S_{j+1} \neq \varnothing$, condition (b) implies that $S_j\cap S_{j+1}$ separates S_j-S_{j+1} and $S_{j+1}-S_j$. Because $x_{j-1} \in S_j-S_{j+1}$ and $x_{j+1} \in S_{j+1}-S_j$, $S_j\cap S_{j+1}$ separates x_{j-1} and x_{j+1}. Consider now the sequence $\{x_{j+1}, \cdots, x_m, x_1, \cdots, x_{j-1}\}$. Clearly there must exist two consecutive nodes x_h and x_{h+1} that are separated by $S_j\cap S_{j+1}$. Because $\{x_h, x_{h+1}\} \subseteq S_{h+1}$, $S_j\cap S_{j+1}$ splits S_{h+1}. This contradicts property (a), and therefore $LHS(M)$ must be β-acyclic.

We can again apply Algorithm 1 to M'', where M'' is obtained from M by removing nonessential keys and by padding the set to make it full. By the proof of Theorem 1, the resulting database scheme **R** is β-acyclic. If **R** is γ-acyclic, we are done. Suppose therefore that **R** is γ-cyclic; we will complete the proof by deriving a contradiction of property (b). We first give a lemma that allows us to restrict our attention to short γ-cycles.

Lemma 4. A hypergraph **R** is γ-acyclic if and only if it is β-acyclic and has no γ-cycles of length 3.

Proof. The only-if direction is immediate, so it is sufficient to prove the if direction. Suppose therefore that **R** is β-acyclic but γ-cyclic. We claim that **R** has a γ-cycle of length 3. Let $(R_1, x_1, \cdots, R_n, x_n, R_{n+1})$ be the shortest γ-cycle in **R**, and suppose that $n \geq 4$. We know that $x_n \in R_i$ for some $1<i<n$, otherwise the cycle would be a weak β-cycle. Since $n \geq 4$, either $n-i+1 \geq 3$ or $i \geq 3$, or else we would have $n<n+1=(n-i+1)+i \leq 2+2=4$. But then either $(R_1, x_1, \cdots, R_i, x_n, R_1)$ or $(R_i, x_i, \cdots, R_n, x_n, R_i)$ is a γ-cycle, contradicting the minimality of n. Therefore the shortest γ-cycle of **R** must be of the form $(R_1, x_1, R_2, x_2, R_3, x_3, R_1)$, where the intersections of the sets are shown in Figure 9. □

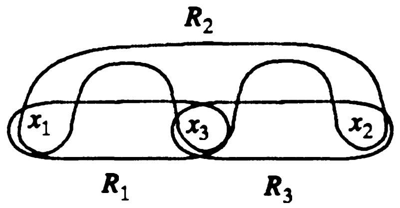

Figure 9. The shortest γ-cycle.

Proof of Theorem 2 continued. Consider the set M^T of MVDs corresponding to a join tree for **R**. Let $P \twoheadrightarrow Q$ be an arbitrary element of M^T, and let x_1, x_2, x_3 and R_1, R_2, R_3 be as in Figure 9. If $x_1 \notin P$ and $x_2 \notin P$, then x_1 and x_2 are connected by R_2-P, and therefore in the same connected component after the P-edge is removed from the join tree. Therefore $P \twoheadrightarrow Q$ cannot separate x_1 and x_2 in this case. If $x_1 \in P$ or $x_2 \in P$, $P \twoheadrightarrow Q$ cannot separate x_1 and x_2 by the definition of separation. Since $P \twoheadrightarrow Q$ was arbitrary, it follows that M^T does not separate x_1 and x_2.

On the other hand, because $R_1 \cap R_3 \neq \varnothing$, Lemma 1 implies that some X in $LHS(M^T)$ separates $R_1 - R_3$ and $R_3 - R_1$. In particular, X separates x_1 and x_2. This contradicts the observation above that M does not separate x_1 and x_2. The counter assumption is therefore false and $\mathbf{R}$ is γ-acyclic. □

For Berge-acyclic join dependencies Proposition 6 gives a good hint: no key in the equivalent set of MVDs should be of size greater than 1. This is stated in condition (a) of Theorem 3. Since key-splitting is prohibited by this condition, the assumption about conflict-freedom can be relaxed into condition (b) of the theorem.

Theorem 3. A set of multivalued dependencies M' is equivalent to a Berge-acyclic join dependency $\bowtie\mathbf{R}$ if and only if M' has a cover M with the following properties:

(a) $|X| \leq 1$ for any X in $LHS(M)$; and

(b) if $A \twoheadrightarrow Y$ and $B \twoheadrightarrow Y$ for some distinct A and B in $LHS(M)$, then $\varnothing \twoheadrightarrow Y$.

Proof. *Only if.* Suppose that $M' \equiv \bowtie\mathbf{R}$ where $\mathbf{R}$ is Berge-acyclic; we must show that M' has a cover M satisfying (a) and (b). Let again M be the set of MVDs that corresponds to a join tree for $\mathbf{R}$.

Suppose that $|X| > 1$ for some X in $LHS(M)$. Since $X = R_1 \cap R_2$ for some R_1, R_2 in $\mathbf{R}$, we have $|R_1 \cap R_2| > 1$. Proposition 6 then implies that $\mathbf{R}$ is Berge-cyclic, contradicting our assumption. Thus (a) holds. Condition (b) follows easily: we know that M is conflict-free (by Proposition 4), and (b) is just a consequence of the condition $DEP(X) \cap DEP(Y) \subseteq DEP(X \cap Y)$ under restriction (a).

If. Suppose that $M' \equiv M$, where M satisfies (a) and (b). We must show that there exists a Berge-acyclic join dependency $\bowtie\mathbf{R}$ such that $M' \equiv \bowtie\mathbf{R}$.

It is easily seen that conditions (a) and (b) of the theorem imply conditions (a) and (b) of Theorem 2. Let $\mathbf{R}$ be the scheme produced by applying Algorithm 1 to M'', where again M'' is obtained from M by removing nonessential keys and by padding it into a full set. By the proof of

Theorem 2, **R** is γ-acyclic.

Suppose then that **R** is Berge-cyclic, and let $(S_1, x_1, \cdots, S_m, x_m, S_{m+1})$ be the shortest Berge-cycle in **R**. If $m \geq 3$, each x_i can only belong to S_i and S_{i+1}, and to no other S_j (otherwise we could delete from the cycle the part between S_j and S_i and obtain a shorter Berge-cycle). But then $(S_1, x_1, \cdots, S_m, x_m, S_{m+1})$ would be a weak β-cycle, contradicting the fact that **R** must be γ-acyclic (and thus β-acyclic).

Therefore $m=2$, and the shortest Berge-cycle is of the form $(S_1, x_1, S_2, x_2, S_1)$. But now $\{x_1, x_2\} \subseteq S_1 \cap S_2$. Any join tree for **R** must therefore contain an edge labelled with X such that $\{x_1, x_2\} \subseteq X$. By Propositions 2 and 4 and by Lemma 1, $X \in LHS(M') \subseteq LHS(M)$. This contradicts condition (a) and proves the claim. □

4. Conclusions

We have characterized acyclic join dependencies of various degree in terms of equivalent sets of multivalued dependencies. The characterizations emphasize the role of the left sides of the dependencies. This is not surprising considering the importance of left sides in the decomposition process [Lie82, GrR83]. The characterizations can be stated in a relatively simple form. Thus they should be helpful to a database designer aiming at an acyclic scheme of a certain degree.

An interesting question that we have not dealt with is what to do if the dependency set does *not* satisfy the necessary conditions. If also functional dependencies are considered, it is sometimes possible to transform a cyclic scheme into an equivalent acyclic scheme [Sac83]. Even if this does not totally succeed, it may be possible to use transformations for localizing the cyclic portion of a scheme [GyP84, Gys85]: this helps, for instance, in using the acyclic part of the scheme more efficiently.

Often it is desirable to achieve a totally acyclic scheme. The solution proposed most often is to modify the set of attributes. Besides acyclicity [Ull82b, GoS82, BeK83], this approach can be used to obtain independence [Sci83]. Our results give further hints on how to do such transformations. For instance, higher degrees of acyclicity can be obtained by splitting some attributes which have been recognized as critical by the characterizations, or by adding some MVDs like in the example in connection with Theorem 2. These questions, however, require further studies.

Acknowledgement

We thank the referees for their detailed comments that helped us to clarify the presentation.

References

ABU79 A.V. Aho, C. Beeri, J.D. Ullman, The theory of joins in relational databases. *ACM Transactions on Database Systems 4*, 3 (September 1979), 297-314.

ADM85 G. Ausiello, A. D'Atri, M. Moscarini, Chordality properties on graphs and minimal conceptual connections in semantic data models. Proceedings of the Fourth ACM SIGACT-SIGMOD Symposium on Principles of Database Systems, March 1985, 164-170.

BBG78 C. Beeri, P.A. Bernstein, N. Goodman, A sophisticate's introduction to database normalisation theory. Proceedings of the Fourth International Conference on Very Large Data Bases, September 1978, 113-124.

BFMY83 C. Beeri, R. Fagin, D. Maier, M. Yannakakis, On the desirability of acyclic database schemes. *J. ACM 30*, 3 (July 1983), 479-513.

BeK83 C. Beeri, M. Kifer, Elimination of intersection anomalies from database schemes. Proceedings of the Second ACM SIGACT-SIGMOD Symposium on Principles of Database Systems, March 1983, 340-351.

DAM84 A. D'Atri, M. Moscarini, On the recognition and design of acyclic databases. Proceedings of the Third ACM SIGACT-SIGMOD Symposium on Principles of Database Systems, April 1984, 1-8.

Fag77 R. Fagin, Multivalued dependencies and a new normal form for relational databases. *ACM Transactions on Database Systems 2*, 3 (September 1977), 262-278.

Fag83a R. Fagin, Degrees of acyclicity for hypergraphs and relational database schemes. *J. ACM 30*, 3 (July 1983), 514-550.

Fag83b R. Fagin, Acyclic database schemes (of various degrees): a painless introduction. *CAAP'83*, 8th Colloquium on Trees in Algebra and Programming, G. Ausiello, M. Protasi (eds.), Springer-Verlag, Berlin - Heidelberg - New York, 1983, 65-89.

FMU82 R. Fagin, A.O. Mendelzon, J.D. Ullman, A simplified universal relation assumption and its properties. *ACM Transactions on Database Systems 7*, 3 (September 1982), 343-360.

GoS82 N. Goodman, O. Shmueli, Transforming cyclic schemas into trees. Proceedings of the ACM Symposium on Principles of Database Systems, March 1982, 49-54.

GoS83 N. Goodman, O. Shmueli, Syntactic characterization of tree database schemas. *J. ACM 30*, 4 (October 1983), 767-786.

GoT84 N. Goodman, Y.C. Tay, A characterization of multivalued dependencies equivalent to a join dependency. *Information Processing Letters 18* (June 1984), 261-266.

Gra79 M.H. Graham, On the universal relation. Technical Report, CSRG, University of Toronto, September 1979.

GrR83 G. Grahne, K.-J. Räihä, Database decomposition into fourth normal form. Proceedings of the Ninth International Conference on Very Large Data Bases, M. Schkolnick, C. Thanos (eds.), October-November 1983, 186-196.

Gys85 M. Gyssens, Embedded join dependencies as a tool for decomposing full join dependencies. Proceedings of the Fourth ACM SIGACT-SIGMOD Symposium on Principles of Database Systems, March 1985, 205-214.

GyP84 M. Gyssens, J. Paredaens, On the decomposition of join dependencies. Proceedings of the Third ACM SIGACT-SIGMOD Symposium on Principles of Database Systems, April 1984, 143-152.

Kat84 H. Katsuno, An extension of conflict-free multivalued dependency sets. *ACM Transactions on Database Systems 9*, 2 (June 1984), 309-326.

Lie81 Y.E. Lien, Hierarchical schemata for relational databases. *ACM Transactions on Database Systems 6*, 1 (March 1981), 48-69.

Lie82 Y.E. Lien, On the equivalence of database models. *J. ACM 29*, 2 (April 1982), 333-362.

Mai83 D. Maier, *The Theory of Relational Databases*. Computer Science Press, Rockville, Md., 1983.

Sac83 D. Sacca', On the recognition of coverings of acyclic database hypergraphs. Proceedings of the Second ACM SIGACT-SIGMOD Symposium on Principles of Database Systems, 1983, 297-304.

SMM84 D. Sacca', F. Manfredi, A. [illegible]a, Properties of database schemata with functional dependencies. Proceedings of the Third ACM SIGACT-SIGMOD Symposium on Principles of Database Systems, April 1984, 19-28.

Sci81 E. Sciore, Real-world MVD's. Proceedings of the ACM SIGMOD 1981 Conference on Management of Data, April 1981, 121-132.

Sci83 E. Sciore, Improving database schemes by adding attributes. Proceedings of the Second ACM SIGACT-SIGMOD Symposium on Principles of Database Systems, 1983, 379-382.

TaY84 R.E. Tarjan, M. Yannakakis, Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. *SIAM Journal on Computing 13*, 3 (August 1984), 566-579.

Ull82a J.D. Ullman, *Principles of Database Systems (Second Edition)*. Computer Science Press, Rockville, Md., 1982.

Ull82b J.D. Ullman, The U.R. strikes back. Proceedings of the ACM Symposium on Principles of Database Systems, March 1982, 10-22.

YuO79 C.T. Yu, M.Z. Ozsoyoglu, An algorithm for tree-query membership of a distributed query. Proceedings of the IEEE COMPSAC'79 Conference, 1979, 306-312.