Characterization of Desirable Properties of General Database Decompositions

Stephen J. Hegner
Department of Computer Science and Electrical Engineering
Votey Building
University of Vermont
Burlington, VT 05405 U. S. A.
Telephone: (802)656-3330
Internet: hegner@uvm.edu

Keywords: database theory, relational databases, decomposition, acyclicity

This paper appears in a special issue on database theory of the *Annals of Mathematics and Artificial Intelligence*, **7**(1993), pp. 129-195.

ABSTRACT

The classical theory of acyclicity of universal relational schemata identifies a set of "desirable" properties of such schemata, and then shows that all of these properties are equivalent to one another, and in turn equivalent to certain acyclicity characterizations of a hypergraph underlying the schema. The desirable properties include the simplicity of constraints, the correctness of certain efficient query evaluation algorithms, and the complexity of maintaining the integrity of a decomposed database.

The principal result of this paper is to show that the essence of this result may be extended to a much more general setting; namely, that in which database schemata are just sets and database mappings just functions. Rather than identifying a single "desirability" class, our work shows that there are several, all of which collapse to a common group when restricted to the universal relational setting. Particularly, the classical notions of "pairwise consistency implies global consistency" and "hypergraph acyclicity" are not equivalent in the general case, but rather are independent of each other, and may be considered separately or in combination, to yield varying strengths of desirability.

0. Introduction

0.1 Motivation and overview

Ever since its introduction a little over twenty years ago, the relational model has been a cornerstone of both research and development in the field of databases. The reasons are not difficult to understand. On the one hand, its simple mathematical structure lends itself well to precise formalization of important concepts, and on the other hand, that same simplicity also contributes to the ease with which simple yet useful user interfaces may be designed. Despite these advantages, it is well known that the relational model is not well suited to all applications. For this reason, a number of newer, more sophisticated data models, such as object-oriented models [1], other so-called semantic data models [21], as well as extensions of the relational model to contexts including nesting [28, Ch. 7], deduction [7], and incomplete information [13],[29], have been introduced and studied in recent years. While there is certainly some supporting theory evolving for these other data models, with the exception of certain aspects of deductive query processing, it may safely be said that the extent of theoretical support for these newer models is nowhere near what it is for the traditional relational model. Yet, the need for such theory can hardly be argued. If these newer models are to enjoy the success that the relational model has, appropriate models of their structure and behavior must be developed.

It is a major thesis of our research that the development of theoretical foundations for other data models should not proceed purely within the context of a single data model. On the contrary, even the most diverse data models share some basic properties, and an understanding of that common ground can provide the foundation upon which the formulation and development of more specific properties is based. Furthermore, the relational model, with its rich collection of theoretical results, can often provide the guidelines along which these more general results may be discovered. In this paper, we substantiate this thesis by taking one of the central results of relational database theory, and extending it to a very general framework which includes virtually all reasonable data models. Specifically, we generalize the classical theory of *acyclic decompositions* of universal relational schemata. In that decomposition theory, numerous desirable properties, including the simplicity of constraints, the correctness of certain efficient query evaluation algorithms, and the complexity of maintaining the integrity of a decomposed database of a single relation, are all shown to be equivalent to the acyclicity of a hypergraph naturally defined by the schema [10],[4]. To aid us in explaining further the nature of our generalization, it is appropriate to begin by recalling the original characterization theorem for acyclic schemata in the universal relational context, as stated and proved in [4, Thm. 3.4].[1]

[1]It is impractical to provide the technical background necessary to understand this result here in the introduction. Readers who are unfamiliar with this aspect of relational database theory are strongly encouraged to read at least the first three sections of the paper [4] to gain an appreciation of its significance. Nonetheless, the technical background necessary to understand this result will be presented in this paper before it is actually used in the development of our generalized results.

0.1.1 Theorem – the classical characterization of acyclicity *The following properties are equivalent on a universal relational schema* $\mathbf{R}$ *with single relation symbol* R *over attribute set* U*, regarded as being decomposed into the set of projections* $\{R_{U_1}[U_1], \ldots, R_{U_n}[U_n]\}$*, with* $\bigcup_{i=1}^{n} U_i = U$*.*

(S1) *Every pairwise consistent* n*-tuple of relations* $(r_1, \ldots, r_n)$ *on* $R_{U_1}[U_1] \times \ldots \times R_{U_n}[U_n]$ *is globally consistent.*

(S2) $\mathbf{R}$ *has a join tree.*

(S3) $\mathbf{R}$ *has the running intersection property.*

(S4) $\mathbf{R}$ *has a monotone join expression.*

(S5) $\mathbf{R}$ *has a sequential monotone join expression.*

(S6) *The join dependency* $\bowtie[U_1, U_2, \ldots, U_n]$ *is equivalent to a set of multivalued dependencies.*

(S7) *The join dependency* $\bowtie[U_1, U_2, \ldots, U_n]$ *is equivalent to a set of conflict-free multivalued dependencies.*

(S8) $\mathbf{R}$ *has a full reducer.*

(S9) *The underlying hypergraph of* $\mathbf{R}$ *is acyclic.*

(S10) *The underlying hypergraph of* $\mathbf{R}$ *is closed-acyclic.*

(S11) *The underlying hypergraph of* $\mathbf{R}$ *is chordal and conformal.*

(S12) *Graham's algorithm succeeds with input* $\mathbf{R}$*.* □

Our generalization of this result is formulated in the most general context possible — database schemata are sets of states (with no further structure required), and database morphisms are just functions between these sets. Since virtually any reasonable data model has states (legal databases) and database mappings between states, the generalization is universally applicable.

Unlike the classical relational result, we cannot provide an equivalence among twelve generalized properties. Rather, we have discovered that there are two distinct "base" classes of equivalent properties of decompositions, which happen to coalesce within the universal relational model under the constraint of only the join dependency defining the decomposition. One such class is founded upon *acyclicity*, which deals with the structure of an underlying hypergraph. Property (S9) in the above theorem is the prototypical example of such a characterization. The other base class of equivalent properties is centered about *pairwise definability*, which recaptures the idea that "two at a time is enough" — consistency of a decomposed database may be checked by independently examining decomposed relations in groups of two. Item (S1) in the above theorem is the most basic example of such a property.

As we shall see, even in the context of a universal relational schema, once we add a few functional dependencies, an acyclic schema no longer need satisfy (S1), and we can find a schema with three relations, constrained only by cross dependencies, which is pairwise

definable but not acyclic in our generalized sense.[2] Furthermore, certain properties, notably the running intersection property (S3) and the join expression properties (S4) and (S5), have natural generalizations which are equivalent to requiring both acyclicity and pairwise definability. The only characterization of the classical theorem which does not fit into one of these three groups is the property (S8) of having a full reducer, which, by its very nature, can be generalized only in a context in which there is an order structure on the database states. In summary, we will show that there is a hierarchy of types for database schema decompositions, as depicted in Figure 0.1.1, with the lower (in the diagram) types strict subsets of the lower ones.

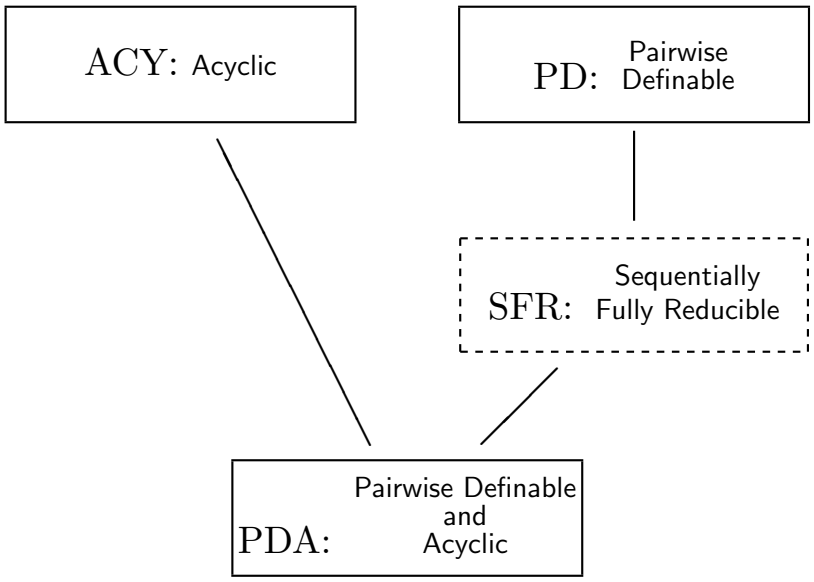


Figure 0.1.1: The hierarchy of types of equivalence for database schema decompositions

The entry labelled "Sequentially Fully Reducible" is the special one which generalizes the notion of full reducer. It is shown in a dashed box because it makes sense in a somewhat less general framework than the others. Since in addition to requiring a framework with order, none of the other characterizations in 0.1.1 generalize to a similar property, we have not earmarked it as a "base" class. But it is nonetheless interesting to note that it lies properly between pairwise definability and the combined concepts of acyclicity and pairwise definability, and so is not equivalent to any of the other classes.

There is another interesting aspect to this classification. The property of acyclicity, by itself, is not very interesting in the general context in which we work. A general decomposition can have an acyclic hypergraph and still not be desirable in any reasonable sense.[3]

[2]The reader who is familiar with relational database theory is encouraged to examine the examples of 3.2.2 through 3.2.4 at this time. Even if some of the technical terms specific to our presentation are not familiar, it should be possible to get the general idea of how fragile 0.1.1 is, relative to adding even a few functional dependencies to the schema.

[3]The reader who is familiar with relational database theory is encouraged to examine 3.2.2 to see how

0.1.1 concept	PD version	ACY version	PDA version	SFR version
(S1)	•			
(S2)		•	•	
(S3)		•	•	
(S4)	•	•	•	
(S5)	•	•	•	
(S6)	•			
(S7)	•			
(S8)				•
(S9)		•		
(S10)		•		
(S11)		•		
(S12)		•		

Figure 0.1.2: Generalizations of conditions of Theorem 0.1.1

But this is not in contradiction to the classical result. After all, in the classical theory, the characterizations (S9) through (S12) of 0.1.1 are not really "desirable properties;" rather, they are abstract properties which guarantee the existence of desirable properties in that context. It is (S1) through (S8) which are truly the "desirable properties," and these we preserve in our generalization.

It is, however, not appropriate to discard hypergraph acyclicity as a useful property in the general context. Rather, it assumes the rôle of augmenting pairwise definability to gurarantee even more desirable properties than pairwise definability can alone. In essence then, ignoring for the moment the special case of a pairwise reducer, there are two useful generalizations of desirability which we extract from the classical setting: the one which considers pairwise definability alone, and the second which considers both pairwise definability and acyclicity. It just happens that in the classical universal relational setting these classes all coincide.

The table of Figure 0.1.2 provides further specific information on the nature of our generalization. The label for each column corresponds to the abbreviation for the equivalence class identified in 0.1.1. A "•" in a position of the table means that the associated universal relational property identified in 0.1.1 has a generalization of the type identified at the top of the column. Notice that some properties have more than one generalization, all of which collapse to the same thing in the case of a simple universal relational schema. Observe also that properties (S9) through (S12) have only one generalization, relating only to acyclicity. This is because they refer only to the hypergraph of the decomposition associated with the schema, and not to any "database" properties. In essence, once we go beyond the universal relational context of 0.1.1, the hypergraph of a decomposition simply does not encode

easy it is to come up with an example to confirm this claim. Essentially, all that it takes is one functional dependency and one join dependency.

enough information by itself to make substantial conclusions about desirable properties of the decomposition. (See Section 3.1 for the definition of the hypergraph of a general schema.)

For more detail, the reader may wish to take a look at the classification and theorem of Section 5.1 before (and while) reading the core of the paper. Although much of the specific terminology will not make sense until the paper is read, the extent of these results can nonetheless be gleaned from examining the precise statements made there.

The outline of this paper is as follows. In Section 1, we provide the basic foundations of mathematical and database-theoretical concepts upon which this work is based. Then, in the next three sections, we develop the specific properties of the types identified in Figure 0.1.1. In Section 2, we examine the class PD, embodying the notion of pairwise definability. In total, we provide fourteen distinct characterizations, including a generalization of the famous *Chinese Remainder Theorem*. In Section 3, we turn to the characterizations of the classes ACY and PDA, thus encompassing both acyclicity alone and in combination with pairwise definability. Section 4 contains a development of order-based properties of schemata, and then examines the class SFR, generalizing the classical notion of a full reducer. Finally, Section 5 provides conclusions and directions for further research.

0.2 Prerequisites and Notation

We have tried to write this paper so that it is reasonably accessible to mathematicians as well as database theorists. However, our review of traditional database theory notions is strictly limited to what we absolutely need to develop our results. In particular, motivational discussions almost nonexistent, so that some familiarity with the standard notation and terminology of the relational model, as may be found in [23] and [28], will prove extremely helpful in understanding why the issues we address are important. Also, several examples require familiarity with basic relational concepts not explicitly covered in this paper. Specific familiarity with the theory of acyclicity in the relational model ([10],[4] or [23, Ch. 13]) will prove very helpful in understanding the generalizations, but is not absolutely necessary, since our discussion includes reviews of the key ideas.

On the mathematical side, since our domain of models is sets and functions, very little special knowledge is necessary beyond a basic grounding in set theory and logic. However, in several places we do use notation and terminology from certain areas of algebra, notably lattice theory. The book [16] is provides a suitable source of further information. Also, we occasionally use terminology from first-order logic, for which [25] is an appropriate reference.

0.3 Relationship to Other Work

We are not aware of any similar pursuit of the generalization of acyclicity to any other data models. However, the topic of decomposition in a set-based framework has fruitfully been pursued by Bancilhon and Spyratos [3], [2], who examined both update semantics and view independence in this context.

This paper shares a common setting with the paper [17]. While here we study properties of decompositions in which the component schemata have nontrivial constraints connecting

them, [17] examines direct decompositions, in which all views are independent. The paper [18] also deals with decompositions in a general context, but focuses upon the problem of updates rather than schema desirability.

This paper began as the full version of the conference paper [19]. However, in preparing the details of one of the proofs, we discovered an error which makes the results of [19] valid only in a rather restricted context. Therefore, the results presented here, rather than those of the conference version, should be taken as the definitive ones.

We have also changed notation in a few instances from that of our earlier papers. But when we have done so, it has been carefully noted, so that hopefully no confusion will result.

1. Schemata and Decompositions

1.1 The Universal Relational Framework

We begin by briefly recalling essential concepts underlying the relational framework in which the traditional acyclicity theory is presented. None of the material in this subsection is new, but we have found it convenient to introduce some new notation which is useful for our purposes.

1.1.1 Relations and universal relational schemata Let U be a finite set, called the set of *attributes*. With each such attribute we associate an infinite set $\mathsf{dom}(A)$, called the *domain* of A. A *U-tuple* is any function t with domain U and $t(A) \in \mathsf{dom}(A)$ for each $A \in U$. A *U-relation* is any set of U-tuples. If $W \subseteq U$ and t is a U-tuple, then $t[W]$ denotes the restriction of t to W, obtained by restricting its defining function to the domain W. Similarly, if M is a U-relation, then $M[W]$ denotes the W-relation obtained by restricting each tuple of M to W.

An *unconstrained universal relational schema* on U is a single U-ary relational symbol R; that is, a relation symbol whose columns are in bijective correspondence with the members of U. The set U should be thought of as the names of the columns of the relation. The *language* of R, denoted $\mathsf{Lang}(R)$, is the first-order language with equality, no function symbols, and with the single U-ary relational symbol R (in addition to equality). We shall always assume that there is some convenient infinite set of variables available; we shall usually denote such variables by the letter v with a convenient subscript. An *R-atom* of $\mathsf{Lang}(R)$ is any formula of the form $R(v_{i_1}, \ldots, v_{i_{\mathsf{Card}(U)}})$, with each v_j a variable. ($\mathsf{Card}(-)$ is the function which gives the cardinality of its argument.)

A finite set of subsets of U which *cover* U in the sense that $\bigcup \mathbf{U} = U$ is called a *finite attribute schema*. Often, as a convenience of notation, it is preferable to work with a finite attribute schema directly, without first identifying an underlying set of attributes, since that set can always be recovered from the finite attribute schema as $\bigcup \mathbf{U}$. As a notational convenience, throughout this paper, whenever $\mathbf{U}$ is a finite attribute schema, $\overline{\mathbf{U}}$ denotes the underlying set of attributes.

For the purposes of this paper, a *dependency* for R is a sentence in the language of R; that is, a sentence with the property that no variable appears in more than one column of R.

Satisfaction is defined in the usual sense of first-order logic, and if Φ is a set of dependencies, we shall write $\mathsf{Mod}(\Phi)$ to denote the set of all relations which satisfy each dependency of Φ, and $\mathsf{Mod}_f(\Phi)$ to denote all such relations which contain only a finite number of tuples. If Φ and Ψ are sets of dependencies, we write $\Phi \models \Psi$ (resp. $\Phi \models_f \Psi$) precisely in the case that $\mathsf{Mod}(\Phi) \subseteq \mathsf{Mod}(\Psi)$ (resp. $\mathsf{Mod}_f(\Phi) \subseteq \mathsf{Mod}_f(\Psi)$). In the case of dependencies which are universal sentences, these two notions are equivalent [11]. Thus, when we are working in the context of such dependencies, we will often write $\models$ instead of $\models_f$, even though our databases are always finite. Note that each model, as defined above, is a U-relation.

Of particular interest in database theory are *full implicational dependencies*, or *FID*'s (called *ID*'s in [8]). Precisely, an *FID* is a typed sentence of the form

$$(Q)(\alpha_1 \wedge \ldots \alpha_k \Rightarrow \beta) \qquad (fid)$$

in which Q is a sequence of universal quantifiers, each quantified variable occurs in at least one of the α_i's, each α_i is is an R-atom of $\mathsf{Lang}(R)$, and β is either an R-atom in $\mathsf{Lang}(R)$, or else an equality of variables of the form $v_i = v_j$, with each of v_i and v_j occurring in some α_i. If β is an equality, then the above sentence is called a *full equality generating dependency* (*FEGD*), and if it is an R-atom in $\mathsf{Lang}(R)$, then it is called a *full tuple generating dependency* (*FTGD*) [11].

In the context of this paper, the most important class of FID's are the FTGD's known as join dependencies. Let $\mathbf{U}$ be a finite attribute schema. For each $A \in \overline{\mathbf{U}}$, let v_A denote a distinguished variable associated with A, and, for each pair $(A, W) \in \overline{\mathbf{U}} \times \mathbf{U}$, let $v_{(A,W)}$ be a distinguished variable associated with that pair. Then, for each $W \in \mathbf{U}$, let α_W denote the R-atom which contains the variable v_A in the A^{th} position if $A \in W$, and which contains the variable $v_{(A,W)}$ in that position otherwise. The *join dependency* $\bowtie[\mathbf{U}]$ is the dependency given by the following formula.

$$(\forall v_A, v_{(A,W)})((\bigwedge_{W \in \mathbf{U}} \alpha_W) \Rightarrow \alpha_{\overline{\mathbf{U}}}) \qquad (jd)$$

Here the universal quantification is meant to range over all variables occurring in the formula. In words, $M \in \mathsf{Mod}(\bowtie[\mathbf{U}])$ iff whenever $\{M_W \mid W \in \mathbf{U}\}$ is a set of $\mathbf{U}$-relations with the property that for each $W_1, W_2 \in \mathbf{U}$, $M_{W_1}[W_1 \cap W_2] = M_{W_2}[W_1 \cap W_2]$, then there is a $\mathbf{U}$-relation M such that $M[W] = M_W[W]$ for each $W \in \mathbf{U}$.

We shall always regard a *multivalued dependency* as a join dependency on two projections; for our purposes, the multivalued dependency $W \twoheadrightarrow V$ on attribute set U *is* the join dependency $\bowtie[W \cup V, W \cup (U \setminus V)]$.

On occasion, we shall also refer to embedded join dependencies. Specifically, if U is a set of attributes and $\mathbf{W}$ is a finite attribute schema with $\overline{\mathbf{W}} \subseteq U$, the *embedded join dependency* $\bowtie[\mathbf{W}]$ holds on a U-relation M whenever the full join dependency $\bowtie[\mathbf{W}]$ holds on the projection of M to the columns indexed by $\overline{\mathbf{W}}$. (See 1.1.4 for a formal definition of projection.) Embedded join dependencies are not FTGD's, but rather involve existential quantification. See [8] for more details.

By a *universal relational schema* on the finite attribute schema $\mathbf{U}$ (also a $\mathbf{U}$-*universal relational schema*), we mean a pair $\mathbf{R} = (R, \Phi)$ in which R is an unconstrained universal

relational schema on U and Φ is a set of FID's on R. The set of all *finite* U-relations which satisfy each of the constraints in Φ (the *legal databases*) of $\mathbf{R}$ is denoted by $\mathsf{LDB}(\mathbf{R})$. The set of all finite U-relations, without regard to dependency satisfaction, is denoted $\mathsf{DB}(\mathbf{R})$. If Ψ is another set of constraints, we say that Φ and Ψ are *logically equivalent* if $\mathsf{LDB}((R[U], \Phi)) = \mathsf{LDB}((R[U], \Psi))$. By definition, we have that Φ and Ψ are logically equivalent iff $\Phi \models_f \Psi$ and $\Psi \models_f \Phi$.

The *simple universal relational schema on* $\mathbf{U}$ (or the *simple* $\mathbf{U}$*-universal relational schema*) is a $\mathbf{U}$- universal relational schema of the form $\mathbf{R} = (R, \emptyset)$; that is, a schema on attribute set $\bigcup \mathbf{U}$ with no dependencies. This schema is also denoted by $\mathbf{R}\langle\mathbf{U}\rangle$.

1.1.2 Minimal legal extensions Let $\mathbf{U}$ be a finite attribute schema, and let $\mathbf{R} = (R, \Phi)$ be a universal relational schema on $\mathbf{U}$. Given $M \in \mathsf{DB}(\mathbf{R})$, a *legal extension* of M is any $N \in \mathsf{LDB}(\mathbf{R})$ with the property that $M \subseteq N$. N is a *minimal legal extension* if it is an extension, and, for any other legal extension P of M, $N \subseteq P$. It is easy to see that, if it exists, a minimal legal extension must be unique and given by $\bigcap\{N \in \mathsf{LDB}(\mathbf{R}) \mid M \subseteq N\}$. When it exists, we shall denote the minimal legal extension of M by $\widehat{M}$ or $(M)\widehat{}$. We have the following important lemma.

1.1.3 Lemma – existence of minimal extensions *Let* $\mathbf{R} = (R, \Phi)$ *be a universal relational schema on attribute set* U*, and assume further that* Φ *is a set of FTGD's. Then, for any* $M \in \mathsf{DB}(\mathbf{R})$*,* $\widehat{M}$ *exists. Furthermore,* $\widehat{M}$ *uses exactly the same domain values as* M*, in the precise sense that if* $A \in U$ *and* $t \in \widehat{M}$*, then there is a tuple* $s \in M$ *such that* $s[A] = t[A]$.

Proof: The proof makes use of the well-known *chase procedure* [30], [14]. The idea is very simple. If a structure M does not satisfy all of the members of Φ, then there is a FTGD of the form (fid) above and an assignment of domain values to variables such that each α_i is satisfied, but β is not satisfied. But then we simply add the tuple required to satisfy β. We continue on in this fashion until all dependencies are satisfied. Since the sentence (fid) is universally quantified, no new domain values can be introduced, and so the process must terminate in a finite number of steps. □

1.1.4 Projections In this item we work within the context that U is a finite set of attributes, and $\mathbf{R} = (R, \Phi)$ is a universal relational schema on U.

For $W \subseteq U$, the W*-projection* of $\mathsf{LDB}(\mathbf{R})$ is the set of W-relations $\mathsf{LDB}(\mathbf{R})[W] = \{M \mid (\exists N \in \mathsf{LDB}(\mathbf{R}))(M = N[W])\}$. We wish to define a view of $\mathbf{R}$ whose legal databases are precisely $\mathsf{LDB}(\mathbf{R})[W]$. Intuitively, a view has two components: the underlying schema (or view schema) and the view mapping. First of all, it is essential to know that $\mathsf{LDB}(\mathbf{R})[W]$ is describable by a set of dependencies. It has been shown by Fagin [8, Thm. 6.3] that there is a set Ψ of FID's such that $\mathsf{LDB}(\mathbf{R})[W] = \mathsf{Mod}_f(\Psi)$. Usually, we denote such a Ψ by $\pi_W(\Phi)$. (Strictly speaking, $\pi_W(\Phi)$ is not unique, but all candidates are logically equivalent, so there will be no lack of rigor in selecting one as "canonical.") Therefore, letting R_W denote a relation symbol on attribute set W, we may define the universal relational

schema $\mathbf{R}_W = (R_W, \pi_W(\Phi))$. Formally, the *W-projection view* of $\mathbf{R}$ is $\Pi_W = (\mathbf{R}_W, \pi_W)$, with $\mathbf{R}_W$ the schema just defined, and $\pi_W : \mathbf{R} \to \mathbf{R}_W$ the relational calculus query defining the projection.[4] The *underlying function* of this projection, which sends each $M \in \mathsf{LDB}(\mathbf{R})$ to $M[W]$, is denoted $\pi_W{}' : \mathsf{LDB}(\mathbf{R}) \to \mathsf{LDB}(\mathbf{R}_W)$. The schema $\mathbf{R}_W$ is the *underlying schema* of the view Π_W and π_W is the *view mapping*.

It is important to understand in particular the nature of $\Pi_\emptyset = (\mathbf{R}_\emptyset, \pi_\emptyset)$. Assuming that $\mathsf{LDB}(\mathbf{R})$ contains at least one nonempty relation, as well as the empty relation $\emptyset$, $\mathsf{LDB}(\mathbf{R}_\emptyset)$ has two legal states, the empty relation $\emptyset$ and $\{()\}$, the relation containing only the empty tuple. For any $M \in \mathsf{LDB}(\mathbf{R})$,

$$\pi_\emptyset{}'(M) = \begin{cases} \{()\} & \text{if } M \neq \emptyset; \\ \emptyset & \text{if } M = \emptyset. \end{cases}$$

Thus, $\Pi_\emptyset$ distinguishes between empty and nonempty relations, but preserves no further information. If $\mathsf{LDB}(\mathbf{R})$ does not contain the empty relation, then $\Pi_\emptyset$ is a trivial view which provides no information.

One further notational remark is in order. Strictly speaking, the notation for the projection mapping should indicate the attribute set of the domain relation. However, in practice, no confusion can result, and to keep the notation simple, we shall write π_W for any projection to attribute set W. Specifically, if $Y \subseteq W \subseteq U$, then π_Y denotes both the projection $\mathbf{R} \to \mathbf{R}_Y$ and the projection $\mathbf{R}_W \to \mathbf{R}_Y$.

1.2 Views and Decompositions in the Set-Based Context

We now turn to the task of abstracting the essential ideas of database schemata, morphisms, and views. Much of the material in this subsection can also be found in [17] and [18], with a somewhat different emphasis.

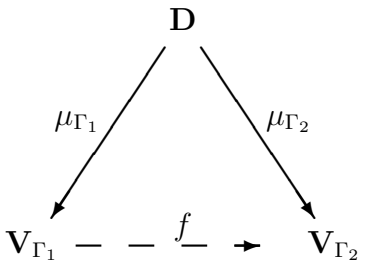
1.2.1 Set-based schemata and views To abstract the notions of the previous paragraph, we discard the information that the schemata are defined by a relation symbol R plus some constraints Φ, and just use the fact that there is an underlying set $\mathsf{LDB}(\mathbf{R})$ of legal states. More precisely, a *set-based database schema* $\mathbf{D}$ (or just a *set-based schema* for short) is a set of *legal databases* (or *legal states*), which we denote by $\mathsf{LDB}(\mathbf{D})$. A *(set based) morphism* $f : \mathbf{D}_1 \to \mathbf{D}_2$ is just a function $f' : \mathsf{LDB}(\mathbf{D}_1) \to \mathsf{LDB}(\mathbf{D}_2)$. A *set-based view* of $\mathbf{D}$ is a pair $\Gamma = (\mathbf{V}_\Gamma, \mu_\Gamma)$ in which $\mathbf{V}_\Gamma$ is a set-based database schema and $\mu_\Gamma : \mathbf{D} \to \mathbf{V}_\Gamma$ is a set-based morphism with the property that $\mu_\Gamma{}' : \mathsf{LDB}(\mathbf{D}) \to \mathsf{LDB}(\mathbf{V}_\Gamma)$ is surjective. *As a matter of notational consistency, unless explicitly stipulated to the contrary (as in projections of a relational schema), given a view with any name* Γ*, the underlying schema shall be denoted by* $\mathbf{V}_\Gamma$ *and the view morphism by* μ_Γ.

The collection of all views of $\mathbf{D}$ is denoted $\mathsf{View}(\mathbf{D})$.

The *congruence* $\mathsf{Congr}(\Gamma)$ of Γ (denoted $\equiv_\Gamma$ in [3]) is the equivalence relation on $\mathsf{LDB}(\mathbf{D})$ defined by $(M_1, M_2) \in \mathsf{Congr}(\Gamma)$ iff $\mu_\Gamma{}'(M_1) = \mu_\Gamma{}'(M_2)$; that is, iff M_1 and M_2 have the

[4] In the terminology of first-order logic, the relational calculus query of the projection is just the defining interpretation formula between the theory of $\mathbf{R}$ and $\mathbf{R}_W$. See [22] for a further discussion of this idea.

same image under the view mapping of Γ. Given set-based views Γ_1 and Γ_2, a *morphism* $f : \Gamma_1 \to \Gamma_2$ of views is a set-based database morphism $f : \mathbf{V}_{\Gamma_1} \to \mathbf{V}_{\Gamma_2}$ such that the following diagram commutes.



It is easy to show [18, 1.2] that there is at most one set-based view morphism $\Gamma_1 \to \Gamma_2$ between two views of the same schema. This morphism exists iff $\mathsf{Congr}(\Gamma_1) \subseteq \mathsf{Congr}(\Gamma_2)$. When it exists, we denote the unique f of the above diagram by $\lambda(\Gamma_1, \Gamma_2)$. This allows us to regard Γ_2 as a view of $\mathbf{V}_{\Gamma_1}$ when $\mathsf{Congr}(\Gamma_1) \subseteq \mathsf{Congr}(\Gamma_2)$. We call this new view of $\mathbf{V}_{\Gamma_1}$ the *relativization* of Γ_2 to Γ_1 and denote it by $\Lambda(\Gamma_1, \Gamma_2) = (\mathbf{V}_{\Gamma_2}, \lambda(\Gamma_1, \Gamma_2))$. The relativization is relational if Γ_1 and Γ_2 are [18, 3.4]; that is, there is a compatible relational structure on Γ_1 such that the morphisms in the above diagram are actually relational database mappings.

If $\Gamma_1 = (\mathbf{V}_{\Gamma_1}, \mu_{\Gamma_1})$ and $\Gamma_2 = (\mathbf{V}_{\Gamma_2}, \mu_{\Gamma_2})$ are set-based views such that there are morphisms $f : \Gamma_1 \to \Gamma_2$ and $g : \Gamma_2 \to \Gamma_1$, then the above uniqueness result guarantees that $g \circ f : \Gamma_1 \to \Gamma_1$ and $f \circ g : \Gamma_2 \to \Gamma_2$ are identity morphisms. Thus, in the standard categorical sense [20, 5.13], f and g are isomorphisms. We say that Γ_1 and Γ_2 are *(set-based) isomorphic* in this case. It is trivial to verify that Γ_1 and Γ_2 are isomorphic iff $\mathsf{Congr}(\Gamma_1) = \mathsf{Congr}(\Gamma_2)$. We write $[\Gamma_1]$ to denote the equivalence class of all views which are (set-based) isomorphic to Γ_1, and $[\mathsf{View}(\mathbf{D})]$ to denote the set of all such equivalence classes.

Upon identifying isomorphic views, view morphism induces a partial order on equivalence classes. As a convenient notation, we write $[\Gamma_2] \leq [\Gamma_1]$ just in case there is a morphism $f : \Gamma_1 \to \Gamma_2$. In an abstract decomposition theory, we do not distinguish between isomorphic views, and, as an abuse of notation, we also write $\Gamma_2 \leq \Gamma_1$, and if both $\Gamma_2 \leq \Gamma_1$ and $\Gamma_1 \leq \Gamma_2$ hold, we may write $\Gamma_1 = \Gamma_2$, even though, strictly speaking, the two views are only isomorphic, so that we should perhaps more properly write $\Gamma_1 \cong \Gamma_2$.

There are two special (equivalence classes of) views in $[\mathsf{View}(\mathbf{D})]$, which are the greatest and least element, respectively. The view whose congruence is the identity relation on $\mathsf{LDB}(\mathbf{D})$ is called the *identity view*, and its canonical representative is denoted $\Gamma_\top(\mathbf{D})$. The view whose congruence is $\mathsf{LDB}(\mathbf{D}) \times \mathsf{LDB}(\mathbf{D})$ is called the *zero view* and a canonical representative is denoted $\Gamma_\bot(\mathbf{D})$. Note that $\Gamma_\bot(\mathbf{D}) \leq \Gamma \leq \Gamma_\top(\mathbf{D})$ for any view $\mathbf{D}$.

1.2.2 Example To make sure that there is no confusion, we illustrate these ideas with a simple relational example. Let $\mathbf{E} = (R[ABC], \{\bowtie[AB, BC], B \to C\})$. (As is customary in relational database theory, we use concatenation of attribute names to denote sets of attributes. Thus, ABC actually means $\{A, B, C\}$, and so forth.) $\mathsf{LDB}(\mathbf{E})$ is just the set of all databases which satisfy the join dependency $\bowtie[AB, BC]$, as well as the functional dependency $B \to C$ [23, Chap. 4]. (A relation satisfies $B \to C$ if its BC projection is

a function.) In Π_{AB}, $\mathbf{E}_{AB}$ is defined by the single relation symbol $R_{AB}[AB]$; there are no nontrivial constraints on this schema. The projective views $\Pi_{BC} = (\mathbf{E}_{BC}, \pi_{BC})$ and $\Pi_B = (\mathbf{E}_B, \pi_B)$ are defined similarly, noting that the functional dependency $B \to C$ is a constraint of $\mathbf{E}_{BC}$. We regard these as set-based schemata and views by "forgetting" the relational structure, and working with the underlying $\mathsf{LDB}(-)$'s. We clearly have that $\Pi_B \leq \Pi_{AB}$ and $\Pi_B \leq \Pi_{BC}$. The morphism $\lambda(\Gamma_{AB}, \Gamma_B)$ is just the projection of R_{AB} onto R_B. In other words, the unique morphism $\Pi_{AB} \to \Pi_B$ just recaptures that we may factor the projection π_B on R_{ABC} through R_{AB}.

1.2.3 The decomposition morphism of a set of views Given a finite attribute schema $\mathbf{U}$ and a $\mathbf{U}$-universal relational schema $\mathbf{R}\langle\mathbf{U}\rangle$, the set $\prod_{W\in\mathbf{U}} \mathsf{LDB}(\mathbf{R})[W]$ is called the set of *local* $\mathbf{U}$*-databases* over $\mathbf{R}\langle\mathbf{U}\rangle$. Given any $(M_W)_{W\in\mathbf{U}} \in \prod_{W\in\mathbf{U}} \mathsf{LDB}(\mathbf{R})[W]$, we know that each M_W is the image under $\pi_W{}'$ of some $M \in \mathsf{LDB}(\mathbf{R}\langle\mathbf{U}\rangle)$; one of the key questions we ask in decomposition theory is whether or not there is a *single* M which is in the inverse image of *each* of the M_W's. This is formalized via the decomposition mapping. Formally, the *decomposition mapping* $\Delta\langle\mathbf{U}\rangle : \mathbf{R}\langle\mathbf{U}\rangle \to \prod_{W\in\mathbf{U}} \mathsf{LDB}(\mathbf{R})[W]$ is defined on elements by $M \mapsto (\prod_{W\in\mathbf{U}} \pi_W{}'(M))$. We think of this mapping as defined by the set $\{\Pi_W \mid W \in \mathbf{U}\}$ of *views*, and generalize this concept to the set-based case as follows. Given a schema $\mathbf{D}$ and a finite set X of views, the *product schema* $\prod_{\Gamma\in X} \mathbf{V}_\Gamma$ is the usual cartesian product, and has as underlying set of states $\prod_{\Gamma\in X} \mathsf{LDB}(\mathbf{V}_\Gamma)$, which we call the *local* X*-databases* over $\mathbf{D}$. The *decomposition morphism* $\Delta\langle X\rangle : \mathbf{D} \to \prod_{W\in\mathbf{U}} \mathbf{R}_W$ is given on elements by $M \mapsto (\mu_{\Gamma_i}{}'(M))_{\Gamma_i\in X}$. We call X a *subdirect decomposition*[5] if $\Delta\langle X\rangle$ is injective. In the case that X consists of just two elements, we call it a *subdirect complementary pair*. The set $\Delta\langle X\rangle'(\mathsf{LDB}(\mathbf{D}))$ is called the set of *decomposed databases*; a subdirect decomposition then is precisely one for which we can recover the original database state from that of the decomposed database. A left inverse of $\Delta\langle X\rangle$ is known as a *reconstruction map*.

In the general set-based case, we cannot speak directly of join dependencies. Rather, we must work with the underlying decomposition, and our generalization of Theorem 0.1.1 will thus make use of properties of subdirect decompositions, rather than properties of join dependencies. To this end, given a join dependency $\bowtie[\mathbf{U}]$, we define the *decomposition of* $\bowtie[\mathbf{U}]$ to be the set $\{\Pi_W \mid W \in \mathbf{U}\}$ of views, which we also denote by $\Pi\mathsf{Views}(\mathbf{U})$. It is immediate that if $\Phi \models \bowtie[\mathbf{U}]$ for the schema $\mathbf{R} = (R[U], \Phi)$, then $\{\Pi_W \mid W \in \mathbf{U}\}$ is a subdirect decomposition of $\mathbf{R}$. As long as we restrict our relational schemata to be constrained by FID's, the converse is also the case, as recorded below.

1.2.4 Proposition *Let* $\mathbf{R}$ *be the universal relational schema* (R, Φ) *on attribute set* $\mathbf{U}$*, with* Φ *a set of FID's. Then* $\{\Pi_W \mid W \in \mathbf{U}\}$ *is a subdirect decomposition of* $\mathbf{R}$ *iff* $\Phi \models \bowtie[\mathbf{U}]$.

PROOF: Consult [31, Cor. 4]. □

In the context of more general classes of dependencies, this relationship breaks down; there exists a decomposition into projections such that the join is not a reconstruction map. See [31, p. 183] for an example.

[5]The terminology is borrowed from universal algebra; see [15, §20] for details.

1.3 Views with Commuting Congruences

In the classical acyclicity characterization, the notion of consistency is central to many of the equivalent characterizations. The idea is that two relations (on different attribute sets) are consistent if they agree on their common columns. In the more general set-based context, we do not have columns. Therefore, we must seek an alternate characterization. In this section, we show how this can be done using views with commuting congruences.

1.3.1 Consistency of views In the context of a universal relational schema $\mathbf{R} = (R, \Phi)$ on finite attribute schema $\mathbf{U}$, let $W_1, W_2 \subseteq \overline{\mathbf{U}}$. The classical definition states that two instances $M_1 \in {\pi_{W_1}}'(\mathsf{LDB}(\mathbf{R}))$ and $M_2 \in {\pi_{W_2}}'(\mathsf{LDB}(\mathbf{R}))$ are *consistent* if they agree on their common columns; *i.e.*, if $\lambda(\Pi_{W_1}, \Pi_{W_1 \cap W_2})'(M_1) = \lambda(\Pi_{W_2}, \Pi_{W_1 \cap W_2})'(M_2)$, or more simply written in light of the comments at the end of 1.1.4, if ${\pi_{W_1 \cap W_2}}'(M_1) = {\pi_{W_1 \cap W_2}}'(M_2)$. The utility of this notion is that only consistent pairs (M_1, M_2) can arise from a common state of $\mathbf{R}$. Note in particular that in the case that $W_1 \cap W_2 = \emptyset$, consistency amounts to agreement on the view $\Pi_\emptyset$; *i.e.*, one projection may be empty iff the other is.

To generalize this idea to the set-based case, let Γ_1, Γ_2, and Γ be arbitrary views of the set-based schema $\mathbf{D}$ with the property that $\Gamma \leq \Gamma_1$ and $\Gamma \leq \Gamma_2$. We say that $M_1 \in \mathbf{V}_{\Gamma_1}$ and $M_2 \in \mathbf{V}_{\Gamma_2}$ are Γ-*consistent* if $\lambda(\Gamma_1, \Gamma)'(M_1) = \lambda(\Gamma_2, \Gamma)'(M_2)$. It is immediate that if $\mathbf{D}$ is the universal relational schema $\mathbf{R}$ identified above, $\Gamma_1 = \Pi_{W_1}$, and $\Gamma_2 = \Pi_{W_2}$, then $\Pi_{W_1 \cap W_2}$-consistency is just consistency in the relational sense, as defined above. However, to complete the generalization, we need to identify a *canonical* Γ which takes the place of $\Pi_{W_1 \cap W_2}$. Unfortunately, this is not possible in general. However, there is a special case which serves our purposes completely, which we develop in 1.3.2 and 1.3.3 below.

In the case that $\mathsf{Congr}(\Gamma_1) \circ \mathsf{Congr}(\Gamma_2) = \mathsf{Congr}(\Gamma_2) \circ \mathsf{Congr}(\Gamma_1)$, with "$\circ$" denoting ordinary relational composition, we say that Γ_1 and Γ_2 have *commuting congruences*, and that $\{\Gamma_1, \Gamma_2\}$ is a *commuting pair*. We now show that this idea generalizes the idea of column intersection for views which are relational projections. First we provide a supporting lemma.

1.3.2 Lemma *Let $\mathbf{R} = (R, \Phi)$ be a $\mathbf{U}$-universal relational schema, and assume further that Φ is a set of FTGD's. Then, for any $W_1, W_2 \in \mathbf{U}$, and $M_1 \in \mathsf{LDB}(\Pi_{W_1})$ and $M_2 \in \mathsf{LDB}(\Pi_{W_2})$ with ${\pi_{W_1 \cap W_2}}'(M_1) = {\pi_{W_1 \cap W_2}}'(M_2)$, there is an $M \in \mathsf{LDB}(\mathbf{R})$ with ${\pi_{W_1}}'(M) = M_1$ and ${\pi_{W_2}}'(M) = M_2$.*

Proof: Pick any $N_1 \in {{\pi_{W_1}}'}^{-1}(M_1)$, subject only to the constraint that for any $A \in W_2 \backslash W_1$, ${\pi_{\{A\}}}'(N_1) \cap {\pi_{\{A\}}}'(M_2) = \emptyset$. In other words, the values in the columns of N_1 labelled by members of $W_1 \setminus W_2$ are disjoint from those of M_2. This is always possible since members of $\mathsf{LDB}(\mathbf{D})$ are finite while domains are infinite. Similarly, pick $N_2 \in {{\pi_{W_2}}'}^{-1}(M_2)$ with ${\pi_{\{A\}}}'(N_2) \cap {\pi_{\{A\}}}'(M_2) = \emptyset$ for all $A \in W_1 \setminus W_2$.

Now, for each $A \in U$, define $\bar{d}(A)$ as follows.

$$\bar{d}(A) = \begin{cases} \mathsf{dom}(A) & \text{if} \quad A \notin W_1 \cup W_2 \\ \mathsf{dom}(A) \cap {\pi_{\{A\}}}'(M_1) & \text{if} \quad A \in W_1 \\ \mathsf{dom}(A) \cap {\pi_{\{A\}}}'(M_2) & \text{if} \quad A \in W_2 \end{cases}$$

Since $\pi_{W_1 \cap W_2}'(N_1) = \pi_{W_1 \cap W_2}'(M_2)$, $\bar{d}(A)$ is well defined. Since Φ is a set of FTGD's, we know by 1.1.3 that $(N_1 \bowtie N_2)^{\frown}$ exists. Now let M be the restriction of $(N_1 \bowtie N_2)^{\frown}$ to just those tuples whose A-domain values for each $A \in U$ lie in $\overline{d}(A)$. Then, since a FTGD is a universal constraint and universal theories are closed under submodels [25, 25.2], $M \in \mathsf{LDB}(\mathbf{R}[\mathbf{U}])$, and, by construction, $\pi_{W_1}'(M) = M_1$ and $\pi_{W_2}'(M) = M_2$. □

Here is the result that assures that the view defined by composing commuting congruences indeed generalizes column intersection in the universal relational case.

1.3.3 Proposition *Let $\mathbf{R} = (R, \Phi)$ be a $\mathbf{U}$-universal relational schema, and assume further that Φ is a set of FTGD's. Then, for any $W_1, W_2 \subseteq \overline{\mathbf{U}}$, Π_{W_1} and Π_{W_2} have commuting congruences, with $\mathsf{Congr}(\Pi_{W_1}) \circ \mathsf{Congr}(\Pi_{W_2}) = \mathsf{Congr}(\Pi_{W_1 \cap W_2})$.*

Proof: Let $(M_1, M_2) \in \mathsf{Congr}(\Pi_{W_1 \cap W_2})$. Then, by the previous lemma, there is an $M \in \mathsf{LDB}(\mathbf{R}[\mathbf{U}])$ with $\pi_{W_1}'(M) = \pi_{W_1}'(M_1)$ and $\pi_{W_2}'(M) = \pi_{W_2}'(M_2)$. But then $(M_1, M) \in \mathsf{Congr}(\Pi_{W_1})$ and $(M, M_2) \in \mathsf{Congr}(\Pi_{W_2})$, so $(M_1, M_2) \in \mathsf{Congr}(\Pi_{W_1}) \circ \mathsf{Congr}(\Pi_{W_2})$. Hence $\mathsf{Congr}(\Pi_{W_1 \cap W_2}) \subseteq \mathsf{Congr}(\Pi_{W_1}) \circ \mathsf{Congr}(\Pi_{W_2})$. But it is immediate that $\mathsf{Congr}(\Pi_{W_1}) \circ \mathsf{Congr}(\Pi_{W_2}) \subseteq \mathsf{Congr}(\Pi_{W_1 \cap W_2})$, so that we have $\mathsf{Congr}(\Pi_{W_1 \cap W_2}) = \mathsf{Congr}(\Pi_{W_1}) \circ \mathsf{Congr}(\Pi_{W_2})$. Similarly, $\mathsf{Congr}(\Pi_{W_1 \cap W_2}) = \mathsf{Congr}(\Pi_{W_2}) \circ \mathsf{Congr}(\Pi_{W_1})$. □

The notion of commuting congruences is of fundamental importance in establishing the various equivalent ideas of schema decomposition. It is also very useful in other contexts. For example, in [18], we show that commuting congruences are of fundamental importance in characterizing the support of view updates. Within that context, we have provided a detailed study, with examples, of the conditions under which the congruences of more general relational views commute. However, the use of commuting congruences actually predates database theory, and arose initially in the context of universal algebra, in an attempt to generalize certain nice properties of subdirect decompositions of groups, rings, and Boolean algebras. Consult [12] and [32] for details.

1.4 Lattice-Like Structures on the Views of a Schema

In 1.2.1, we identified a natural ordering on equivalence classes of views. This order structure gives rise to two distinct lattice-like structures, whose properties we now develop.

1.4.1 The lattice of views of a schema It is well known that, given any set S, the set $\mathsf{EqRel}(S)$ of all equivalence relations on S forms a bounded lattice [16, Ch. IV, Sec. 4, Lemma 1].[6] The join $\rho_1 \vee \rho_2$ of two equivalence relations ρ_1 and ρ_2 is just their intersection, while the meet $\rho_1 \wedge \rho_2$ is the finest equivalence relation containing both ρ_1 and ρ_2. The greatest element of the lattice is the diagonal relation $\mathsf{Diag}(S) = \{(s, s) \mid s \in S\}$, while the least element is just $S \times S$. (We say that a lattice is *bounded* precisely in the case that it has a greatest and least

[6] Our definition is actually the dual of the definition given [16], with meet and join reversed. We do this for compatibility with with the corresponding relational notions. Also, Grätzer couches his definitions in terms of partitions rather than equivalence relations, but the two notions are completely equivalent.

element.) The induced partial order is just reverse inclusion; that is, $\rho_1 \leq \rho_2$ iff $\rho_2 \subseteq \rho_1$. We denote the lattice of all equivalence relations on S by $\mathbf{EqRel}(S) = (\mathsf{EqRel}(S), \vee, \wedge, \top, \bot)$. (Usually, when no confusion can result, we use $\top$ and $\bot$ without subscripts to denote the greatest and least element of a bounded structure, respectively.)

In 1.2.1, we noted that each class of isomorphic views of a set-based schema $\mathbf{D}$ may be identified with the congruence of any of its members. But congruences on set-based views of $\mathbf{D}$ are nothing more than equivalence relations on $\mathsf{LDB}(\mathbf{D})$, and so the set of all equivalence classes of views of $\mathbf{D}$ may be regarded as a bounded lattice. We shall denote this lattice by $[\mathbf{View}(\mathbf{D})] = ([\mathsf{View}(\mathbf{D})], \vee, \wedge, [\Gamma_\top(\mathbf{D})], [\Gamma_\bot(\mathbf{D})])$. Clearly, the two lattices $[\mathbf{View}(\mathbf{D})]$ and $\mathbf{EqRel}(\mathsf{LDB}(\mathbf{D}))$ are naturally isomorphic. As an abuse of notation, we sometimes drop the "$[-]$" equivalence class notation when dealing with elements, and write, *e.g.*, $\Gamma_1 \vee \Gamma_2$ when we really mean $[\Gamma_1] \vee [\Gamma_2]$.

It is important to realize that this construction is guaranteed to yield a lattice only in the set-based case. For example, if Γ_1 and Γ_2 are relational views of a relational schema, there is no guarantee that $\Gamma_1 \vee \Gamma_2$ or $\Gamma_1 \wedge \Gamma_2$ will be representable as a relational schema.

Intuitively, the join $\Gamma_1 \vee \Gamma_2$ provides a view which contains all of the information about the base schema which is contained in either of the views; in effect, it provides a sort of union of the information. In a relational context, it is often represented by the join operation. Specifically, if we have a relational schema $\mathbf{R} = (R, \Phi)$ and two projections Π_{W_1} and Π_{W_2}, then $\Pi_{W_1} \vee \Pi_{W_2} = \Pi_{W_1 \cup W_2}$ iff $\Phi \models \bowtie [W_1, W_2]$; that is, iff the (relational)) join of the W_1-projection and the W_2-projection is the $(W_1 \cup W_2)$-projection. So here we see that the relational join and the lattice-theoretic join are one and the same!

On the other hand, the meet $\Gamma_1 \wedge \Gamma_2$ provides a view which contains the information common to the two views Γ_1 and Γ_2. As already shown in 1.3.3, in the relational context, it is often represented as column intersection. Specifically, in the above context, $\Pi_{W_1} \wedge \Pi_{W_2} = \Pi_{W_1 \cap W_2}$ provided that Φ is a set of FTGD's. On the other hand, if Φ contains other types of dependencies, this relationship may not hold. See [18, 1.11] for an example of how functional dependencies (which are FEGD's and not FTGD's) can cause this relationship to fail.

The critical property in the definition of meet is the commutativity of the corresponding congruences. Indeed, as we shall see in Section 2.1 of this paper, commuting congruences play a critical rôle in characterizing desirable decompositions. Indeed, it is so central to the definition of meet that we define a special lattice-like structure on the views of a schema $\mathbf{D}$ which uses this property.

1.4.2 The bounded weak partial lattice of views In the lattice of views, taking the join of two views Γ_1 and Γ_2 corresponds to taking the least upper bound of their congruences; that is, to finding the coarsest equivalence relation which is finer than both congruences. Since the intersection of two equivalence relations on the same set is itself an equivalence relation on that set – and is in fact the least upper bound of the two arguments – the notion of join is simple and straightforward to define. There would seem to be no other appropriate definition. On the other hand, the notion of meet is not so simple. Taking the meet of two views Γ_1 and Γ_2 corresponds to finding the greatest lower bound of their congruences; that

is, to finding the finest equivalence relation which is coarser than the congruence of each. Unfortunately, the union of two equivalence relations need not be an equivalence relation, as it may fail to be transitive. This necessitates, in effect, a two-step process for computing the meet. First we take the union of the two argument congruences, and then we close up the result under transitivity. This closure under transitivity may be computed by iterating the composition

$$\mathsf{Congr}(\Gamma_1) \circ \mathsf{Congr}(\Gamma_2) \circ \mathsf{Congr}(\Gamma_1) \circ \mathsf{Congr}(\Gamma_2) \circ \mathsf{Congr}(\Gamma_1) \ldots$$

as many times as necessary. (Although to show that any given pair (a, b) is in the transitive closure of $\mathsf{Congr}(\Gamma_1) \cup \mathsf{Congr}(\Gamma_2)$ it suffices to consider a certain finite number of iterations, there may be no single bound which works for all such pairs.) In other words, the meet may be a relatively complex entity to compute. However, if the two congruences commute, this is, if $\mathsf{Congr}(\Gamma_1) \circ \mathsf{Congr}(\Gamma_2) = \mathsf{Congr}(\Gamma_2) \circ \mathsf{Congr}(\Gamma_1)$, then we may rearrange all of the terms in the above iteration to get just $\mathsf{Congr}(\Gamma_1) \circ \mathsf{Congr}(\Gamma_2)$ as the meet. In other words, the greatest lower bound of two equivalence relations is their composition, provided that their congruences commute. (Notice that this property is already suggested by 1.3.3 in the context of relational schemata. See [26] for a further discussion of this topic in a general context.) As we shall see in Section 2.1, this notion of commuting congruences has further important consequences in terms of defining schema simplicity. For that reason, we provide an explicit alternate definition of meet which incorporates this property, as follows.

Given a set S, we define another lattice-like structure on $\mathsf{EqRel}(S)$. The join is the same as that defined in 1.4.1; namely $\rho_1 \vee \rho_2 = \rho_1 \cap \rho_2$. Similarly the greatest and least elements are the same. However, the meet operation is now partial. Specifically, we define $\mathbf{CEqRel}(S) = (\mathsf{EqRel}(S), \vee, \mathbin{\mathbb{\wedge}}, \top, \bot)$ to be the structure which is the same as $\mathbf{EqRel}(S)$, save that the "meet" operation is now defined as follows. (The acronym "CEqRel" stands for "Commuting Equivalence Relations.")

$$\rho_1 \mathbin{\mathbb{\wedge}} \rho_2 = \begin{cases} \rho_1 \wedge \rho_2 & \text{if } \rho_1 \circ \rho_2 = \rho_2 \circ \rho_1; \\ \text{undefined} & \text{otherwise.} \end{cases}$$

We call $\mathbb{\wedge}$ the *strong meet* operator. In general, it is a partial operation, since there are congruences which do not commute. The algebraic structure $\mathbf{CEqRel}(S)$ forms what is known as a *(bounded) weak partial lattice* [16, Ch. I, Sec. 5, Def. 14]. When the operations are defined, the behavior of the operations is just as in a lattice. However, the meet fails to be a total operation.

Now returning to the context of a set-based database schema $\mathbf{D}$, upon invoking the same correspondence between equivalence classes of views and their congruences as identified in 1.4.1, it is easy to translate the structure of $\mathbf{CEqRel}(\mathsf{LDB}(\mathbf{D}))$ to $[\mathsf{View}(\mathbf{D})]$. Specifically, $[\mathbf{CView}(\mathbf{D})] = ([\mathsf{View}(\mathbf{D})], \vee, \mathbin{\mathbb{\wedge}}, [\Gamma_\top(\mathbf{D})], [\Gamma_\bot(\mathbf{D})])$ is the bounded weak partial lattice which is identical to $[\mathbf{View}(\mathbf{D})]]$, save that the meet operation is now $\mathbb{\wedge}$ rather than $\wedge$. In terms of views Γ_1 and Γ_2, this means that

$$[\Gamma_1] \mathbin{\mathbb{\wedge}} [\Gamma_2] = \begin{cases} \{\Gamma \mid \mathsf{Congr}(\Gamma) = \mathsf{Congr}(\Gamma_1) \circ \mathsf{Congr}(\Gamma_2)\} & \text{if } \{\Gamma_1, \Gamma_2\} \text{ is a commuting pair;} \\ \text{undefined} & \text{otherwise.} \end{cases}$$

As with the case of $\wedge$, we will often write $\Gamma_1 \Cap \Gamma_2$ to denote a canonical representative of $[\Gamma_1] \Cap [\Gamma_2]$.[7]

1.4.3 Spans of sets of elements In a lattice-like structure, any set of elements generates a substructure, obtained by closing up that set under specifically identified operations. Specifically, let $\mathbf{D}$ be a set-based schema, and let $X \subseteq [\mathsf{View}(\mathbf{D})]$. We define $\mathsf{Span}(X, [\mathbf{View}(\mathbf{D})])$ to be the smallest subset of $[\mathsf{View}(\mathbf{D})]$ which contains X and which is closed under the operations $\vee$ and $\wedge$. Similarly, we define $\mathsf{Span}(X, [\mathbf{CView}(\mathbf{D})])$ to be the smallest subset of $[\mathsf{View}(\mathbf{D})]$ which contains X and which is closed under the operations $\vee$ and $\Cap$. It is clear that for any X, $\mathsf{Span}(X, [\mathbf{CView}(\mathbf{D})]) \subseteq \mathsf{Span}(X, [\mathbf{View}(\mathbf{D})])$; the reverse inclusion holds only when all constructed congruences commute with one another. It is also clear that $\mathsf{Span}(X, [\mathbf{View}(\mathbf{D})])$ may be regarded as sublattice of $[\mathbf{View}(\mathbf{D})]$ and that $\mathsf{Span}(X, [\mathbf{CView}(\mathbf{D})])$ may be regarded as a weak partial sublattice of $[\mathbf{CView}(\mathbf{D})]$. However, note particularly that we do **not** require that $[\Gamma_\top(\mathbf{D})]$ and $[\Gamma_\bot(\mathbf{D})]$ be part of these sets. Indeed, while each substructure will have as greatest element $\bigvee X$ (because joins of views always exist in the set-based context), $\mathsf{Span}(X, [\mathbf{CView}(\mathbf{D})])$ need not have a least element, (because we are not assured that all congruences will commute).

While the notion of the span of a set X of views will prove invaluable in later characterizations of schema properties, the following immediate characterization of subdirect decomposition is also important.

1.4.4 Proposition — lattice-theoretic characterization of subdirect decomposition *Let $\mathbf{D}$ be a set-based schema, and let X be a set of views of $\mathbf{D}$. Then X is a subdirect decomposition of $\mathbf{D}$ iff $[\Gamma_\top(\mathbf{D})] \in \mathsf{Span}(X, [\mathbf{View}(\mathbf{D})])$.*

PROOF: This is essentially a restatement of a well-known theorem of universal algebra; see, *e.g.*, [15, §20, Thm. 2]. Note that $[\Gamma_\top(\mathbf{D})]$ is the upper bound of the lattice. □

1.4.5 Distributive lattices and sublattices There is one more construction on lattice-like structures which we will need. Let $\mathbf{L} = (L, \vee, \wedge)$ be any lattice. Recall that $\mathbf{L}$ is *distributive* if any triple (a, b, c) of elements from $\mathbf{L}$ satisfies $(a \vee b) \wedge c = (a \wedge c) \vee (b \wedge c)$. If $Y \subseteq L$, we say that Y *is* a distributive sublattice of a lattice or weak partial lattice $\mathbf{L}$ just in case that under the inherited operations, it forms a distributive lattice. In particular, all meets and joins must be defined for all elements of Y, even if $\mathbf{L}$ is only a weak partial lattice.

We shall apply this construction particularly to sets of the form $\mathsf{Span}(X, \mathbf{L})$ in which $\mathbf{L}$ is one of $[\mathbf{View}(\mathbf{D})]$ and $[\mathbf{CView}(\mathbf{D})]$.

[7] In earlier papers [17], [18], [19], used the notation $[\mathbf{View}(\mathbf{D})]$ to denote what we are here calling $[\mathbf{CView}(\mathbf{D})]$, and we used $\wedge$ to denote what we here denote $\Cap$. We did not use the full bounded lattice of views in those earlier works. The new notation seems more reasonable, and hopefully readers of those earlier papers will not become confused because of this change.

1.5 Essentials of Hypergraphs

Hypergraphs play an essential rôle in our work, both in utilizing the classical results to identify the connections between the classical and generalized results, and in establishing the more general results themselves. It is therefore appropriate that we provide a brief yet complete identification of the key results, viewed in a context appropriate for this paper. The reader may find more complete coverage of the database aspects in both the paper [4] and in the text [23], and additional material on hypergraphs in a purely mathematical context in [5].

1.5.1 Basic definitions A *hypergraph* is a pair $\mathbf{H} = (S, E)$ in which S is any finite[8] set (called the set of *nodes*) and E is a subset of $\mathcal{P}(S)$, the set of all subsets of S. The members of E are called the *hyperedges* of $\mathbf{H}$. Hypergraphs generalize ordinary undirected graphs — an ordinary graph is just a hypergraph for which each hyperedge contains exactly two nodes (and is called an *edge*, of course). As a convenient notation, if $\mathbf{H}$ is a hypergraph, we shall let $\mathsf{Nodes}(\mathbf{H})$ denote the nodes of $\mathbf{H}$ and $\mathsf{Edges}(\mathbf{H})$ the hyperedges. Thus, we may also write $\mathbf{H} = (\mathsf{Nodes}(\mathbf{H}), \mathsf{Edges}(\mathbf{H}))$.

It is possible to define the notions *chordal* and *conformal* for hypergraphs as extensions of the definitions usually reserved for ordinary graphs. However, these definitions are rather involved, and since we shall not need to use them directly, we refer the reader to [4] or [5].

1.5.2 The hypergraph of a relational schema While the results presented in this subsection hold on abstract hypergraphs and have no essential connection to database theory, it is nonetheless important to understand how hypergraphs arise in the relational setting. Let $\mathbf{U}$ be a finite attribute schema, and let $\mathbf{R}\langle\mathbf{U}\rangle$ be a simple $\mathbf{U}$-universal relational schema. The *hypergraph* of $\mathbf{U}$ is just $(\cup\mathbf{U}, \mathbf{U})$, and is denoted $\mathcal{R}(\mathbf{U})$. In other words, the attributes are the nodes, and the hyperedges are just the sets of attributes in $\mathbf{U}$. If $\mathbf{R}\langle\mathbf{U}\rangle$ is the simple universal relational schema over $\mathbf{U}$, then the *hypergraph of* $\mathbf{R}\langle\mathbf{U}\rangle$ is just the hypergraph of $\mathbf{U}$. Thus, there is a natural correspondence between simple universal relational schemata and hypergraphs. Indeed, we often regard the simple universal relational schema $\mathbf{R}\langle\mathbf{U}\rangle$ as a hypergraph, as is often done in [4].

1.5.3 Acyclicity of hypergraphs The hypergraph $\mathbf{H}$ is *reduced* if for any two hyperedges $H_1, H_2 \in \mathsf{Edges}(\mathbf{H})$, $H_1 \subseteq H_2 \Rightarrow H_1 = H_2$. Given any hypergraph $\mathbf{H}$, its *reduction* is obtained by discarding each hyperedge which is a subset of another hyperedge. $\mathbf{H}$ is *trivial* if its reduction has only one hyperedge; otherwise it is *nontrivial*.

Given nodes a and b of the hypergraph $\mathbf{H}$, a *path* from a to b is a finite sequence of hyperedges $H_1, \ldots, H_n$ such that $a \in H_1$, $b \in H_n$, and $H_i \cap H_{i+1} \neq \emptyset$ for $1 \leq i \leq n$. In this case, we also say that $H_1, \ldots H_n$ is a path from H_1 to H_n. $\mathbf{H}$ is *connected* if there is a path between any two nodes. Given a set $N \subseteq \mathsf{Nodes}(\mathbf{H})$, the hypergraph *generated* by N

[8]The basic properties of hypergraphs do not depend in any way upon finiteness. However, certain of the constructions which arise in a database context only make sense on finite hypergraphs, and so we shall restrict our attention to that context.

is the reduction of $(N, \{H \cap N \mid H \in \mathsf{Edges}(\mathbf{H})\})$, and is denoted by $\mathsf{HGen}(\mathbf{H}, N)$. A *closed subhypergraph* of $\mathbf{H}$ is any such $\mathsf{HGen}(\mathbf{H}, N)$ with the property that $\mathsf{Edges}(\mathsf{HGen}(\mathbf{H}, N)) \subseteq \mathsf{Edges}(\mathbf{H})$. A *connected component* of $\mathbf{H}$ is a hypergraph generated by a maximal connected set of nodes.

An *articulation pair* for the hypergraph $\mathbf{H}$ is a pair $\{H_1, H_2\} \subseteq \mathsf{Edges}(\mathbf{H})$ of hyperedges such that $\mathsf{HGen}(\mathbf{H}, \mathsf{Nodes}(\mathbf{H}) \setminus (H_1 \cap H_2))$ has strictly more connected components than $\mathbf{H}$ itself. If $S_1, S_2 \subseteq \mathsf{Nodes}(\mathbf{H})$, we say that the articulation pair $\{H_1, H_2\}$ *separates* $\{S_1, S_2\}$ if $S_1 \cap S_2 \subseteq H_1 \cap H_2$. An *articulation set* for $\mathbf{H}$ is any set of nodes of the form $H_1 \cap H_2$, with $\{H_1, H_2\}$ an articulation pair.

A *block*[9] of $\mathbf{H}$ is a set $N \subseteq \mathsf{Nodes}(\mathbf{H})$ such that $\mathsf{HGen}(\mathbf{H}, N)$ has no articulation pair. A block N is *trivial* if it is a subset of some hyperedge of $\mathbf{H}$; otherwise it is *nontrivial*. A hypergraph is *cyclic* if its reduction contains at least one nontrivial block; otherwise it is *acyclic*. A block N is *closed* if $\mathsf{HGen}(\mathbf{H}, N)$ is a closed subhypergraph of $\mathbf{H}$. A hypergraph is *closed cyclic* if it has a nontrivial closed block; otherwise it is *closed acyclic*.

1.5.4 Graham's algorithm There is a procedure known as *Graham's algorithm* which may be applied to a hypergraph $\mathbf{H}$. There are two rules which may be applied in any order, and which generate a new hypergraph from the old one.

(rule-g1) If a node a appears in only one hyperedge, then remove that node from that hyperedge.

(rule-g2) Replace the current hypergraph with its reduction.

Graham's algorithm *succeeds* if it terminates with the empty hypergraph (with no nodes or nonempty edges); otherwise it *fails*.

1.5.5 Intersection graphs and join trees Again let $\mathbf{H}$ be an arbitrary hypergraph. The *complete intersection graph* for $\mathbf{H}$ is the undirected graph which has as nodes the members of $\mathsf{Edges}(\mathbf{H})$ with the edge from E_1 to E_2 labelled by $E_1 \cap E_2$. An *intersection graph* for $\mathbf{H}$ is any subgraph of the complete intersection graph which is obtained by deleting only edges, and not nodes. An *intersection tree* is an intersection graph which is a tree. A *join graph* is an intersection graph with the property that for every $E_1, E_2 \in \mathsf{Edges}(\mathbf{H})$, there is a path from E_1 to E_2 which contains every attribute common to E_1 and E_2 in every edge label on that path. Note that the complete intersection graph is always a join graph. A *join tree* is a join graph which is a tree. (By definition, we take trees to be connected. But note in particular that if $E_1 \cap E_2 = \emptyset$, then the complete intersection graph for $\mathbf{H}$ still has an edge from E_1 to E_2, which is labelled $\emptyset$. Thus, the complete intersection graph is connected (and complete) even if the hypergraph is disconnected, and so a join tree is still a possibility.))

1.5.6 The running intersection property Let $\mathbf{H}$ be an arbitrary hypergraph. An *ordering* for $\mathsf{Edges}(\mathbf{H})$ is any bijection $\sigma : \{1, \ldots, \mathsf{Card}(\mathsf{Edges}(\mathbf{H}))\} \to \mathsf{Edges}(\mathbf{H})$. We say that such a σ is an *articulation ordering* (or *articulation sequence*) if for every i, $1 < i \leq$

[9] In [4], a block is defined to be the set of hyperedges $\mathsf{Edges}(\mathsf{HGen}(\mathbf{H}, N))$ of one of our blocks N.

$\mathsf{Card}(\mathsf{Edges}(\mathbf{H}))$, there is a $j \leq i-1$ with $(\bigcup_{k=1}^{i-1} \sigma(k)) \cap \sigma(i) \subseteq \sigma(j)$. We say that $\mathbf{H}$ has the *running intersection* property if there is an articulation ordering for $\mathsf{Edges}(\mathbf{H})$. (The use of the term *articulation* is motivated by the fact that, in such an ordering, for each edge $\sigma(i)$ with $i > 1$ there is a $j < i$ such that $\{\sigma(i), \sigma(j)\}$ forms an articulation pair for $\mathbf{H}$.) For an articulation ordering σ and for each i, $1 < i \leq \mathsf{Card}(\mathsf{Edges}(\mathbf{H}))$, a $\sigma(j) \in \{\sigma(k) \mid 1 \leq k < i\}$ which satisfies $(\bigcup_{k=1}^{i-1} \sigma(k)) \cap \sigma(i) \subseteq \sigma(j)$ is called an *articulation predecessor* for $\sigma(i)$, and a function $p : \{2, \ldots, \mathsf{Card}(\mathsf{Edges}(\mathbf{H}))\} \to \{1, \ldots, \mathsf{Card}(\mathsf{Edges}(\mathbf{H}))\}$ is called an *articulation predecessor function* for σ provided that $\sigma(p(i))$ is a articulation predecessor of $\sigma(i)$ for each i, $2 \leq i \leq \mathsf{Card}(\mathsf{Edges}(\mathbf{H}))$. For any k, $1 \leq k \leq \mathsf{Card}(\mathsf{Edges}(\mathbf{H}))$, the set $\{\sigma(i) \mid 1 \leq i \leq k\}$ is called an *initial set* for σ.

We are now ready to state the "purely hypergraph-based" equivalences which are part of the classical theorem 0.1.1.

1.5.7 Theorem — equivalence of hypergraph properties *Let* $\mathbf{H}$ *be any hypergraph whatever. Then the following conditions are equivalent.*

(a) $\mathbf{H}$ *is acyclic.*

(b) $\mathbf{H}$ *is closed acyclic.*

(c) $\mathbf{H}$ *is chordal and conformal.*

(d) *Graham's algorithm succeeds on* $\mathbf{H}$.

(e) $\mathbf{H}$ *has a join tree.*

(f) $\mathbf{H}$ *has the running intersection property.*

PROOF: Consult [4]. It is important to understand that while the above theorem is developed and proved in [4] within developed in the context of relational database theory for application to hypergraphs of the form $\mathcal{R}(\mathbf{U})$, it is really a theorem about hypergraphs, and the proof does not depend in any way upon special properties of hypergraphs arising from simple universal database schemata. (Indeed, there are no such properties!) □

The conditions (S4) and (S5) of 0.1.1 are not purely hypergraph-based, because they involve the notion of monotonicity, which only makes sense in the context of ordered database states. However, there is an alternate property of join expressions, termed *articulated*, which can be expressed solely in terms of the supporting hypergraph. Later, in 3.3.6, we shall show that in the relational context, a join plan is monotonic iff it is articulated. For now, we establish the equivalence between acyclicity of a hypergraph and the existence of articulated join plans.

1.5.8 Join plans for hypergraphs Let $\mathbf{H}$ be a hypergraph. A *join plan* for $\mathbf{H}$ is a nonempty binary tree which has the following properties.

(i) Each node has either zero or two children.

(ii) The leaves are labelled with singleton sets of the form $\{H\}$, with $H \in \mathsf{Edges}(\mathbf{H})$. Every such $\{H\}$ labels at least one leaf.

(iii) Each nonleaf node is labelled with the union of the sets labelling its two children. Thus, each node is labelled with a subhypergraph of $\mathbf{H}$.

A join plan is *sequential* if at least one child of each node in the tree is a leaf. A join plan J is *articulated* if for each node n of J, $\mathsf{Label}(n)$ is an initial set of some running intersection sequence of $\mathbf{H}$. (Here $\mathsf{Label}(n)$ denotes the set of hyperedges which label the node n.)

1.5.9 Theorem — equivalence of further hypergraph properties *Let* $\mathbf{H}$ *be any hypergraph whatever. Then the following conditions are equivalent.*

(a) $\mathbf{H}$ *is acyclic.*

(b) $\mathbf{H}$ *has an articulated join plan.*

(c) $\mathbf{H}$ *has an articulated sequential join plan.*

PROOF: We establish the equivalences (a) ⇒ (c), (c) ⇒ (b), and (b) ⇒ (a). Note that (c) ⇒ (b) is trivial, so we must only prove the other two.

First of all, assume condition (a), *i.e.*, that $\mathbf{H}$ is acyclic, and let σ be any articulation ordering for $\mathsf{Edges}(\mathbf{H})$. It is then immediate that the join plan shown in Figure 1.5.1 is articulated and sequential, so that (c) is satisfied. Hence (a) ⇒ (c).

Finally, assume condition (b), that is, that $\mathbf{H}$ has an articulated join plan J. Then, in particular, the root of J, which is labelled with $\mathsf{Edges}(\mathbf{H})$, must admit an articulation ordering, and so by 1.5.7, must be acyclic. □

There is an alternate characterization of articulated join plans which will prove useful. It is presented below, with the support of the following lemma.

1.5.10 Lemma *Let* $\mathbf{H}$ *be a hypergraph. and let* $\sigma : \{1, \ldots, \mathsf{Card}(\mathsf{Edges}(\mathbf{H}))\} \to \mathsf{Edges}(\mathbf{H})$ *be any ordering of the edges of* $\mathbf{H}$*. Then* σ *is an articulation ordering for* $\mathsf{Edges}(\mathbf{H})$ *iff there is a a function* $p : \{2, \ldots, \mathsf{Card}(\mathsf{Edges}(\mathbf{H}))\} \to \{1, \ldots, \mathsf{Card}(\mathsf{Edges}(\mathbf{H}))\}$ *satisfying the following two conditions.*

(i) *For each* i, $1 < i \leq \mathsf{Card}(\mathsf{Edges}(\mathbf{H}))$, $p(i) < i$.

(ii) *The intersection graph* G *for* $\mathbf{H}$ *whose set of edges is exactly that which connects the pairs* $\{\{\sigma(i), \sigma(p(i))\} \mid 1 < i \leq \mathsf{Card}(\mathsf{Edges}(\mathbf{H}))\}$ *is a join tree.*

PROOF: Starting with an articulation ordering σ for $\mathsf{Edges}(\mathbf{H})$, we build an intersection graph T as follows. Let p be defined such that, for each i, $1 < i \leq \mathsf{Card}(\mathsf{Edges}(\mathbf{H}))$, $p(i) < i$ and $(\bigcup_{k=1}^{i-1} \sigma(k)) \cap \sigma(i) \subseteq \sigma(p(i))$, and include in T the edge from $\sigma(i)$ to $\sigma(p(i))$. It is straightforward to show that T is a join tree. Conversely, starting with a join tree T and a function p satisfying (i) and (ii) above, we may build an articulation ordering as follows. Start by picking any hyperedge H of $\mathbf{H}$ whatever, and call it $\sigma(1)$. Now, assume that we have selected $\sigma(1)$ through $\sigma(i)$. If $i < \mathsf{Card}(\mathsf{Edges}(\mathbf{H}))$, then pick $\sigma(i+1)$ to be any node

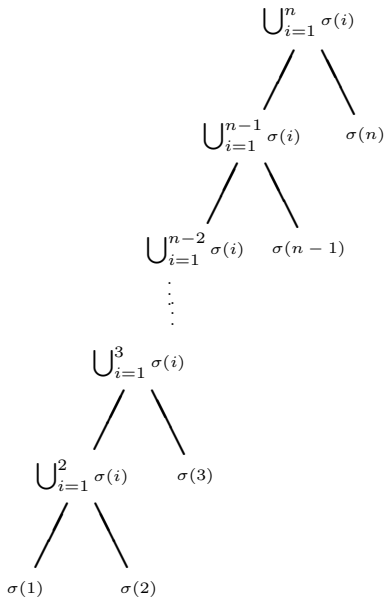


Figure 1.5.1: Sequential join plan for 1.5.9

which is directly connected in T to some node $\sigma(j)$ in $\{\sigma(i) \mid 1 \leq i \leq i\}$, but is not itself in that set. It is straightforward to confirm that $(\bigcup_{k=1}^{i-1} \sigma(k)) \cap \sigma(i) \subseteq \sigma(j)$, and so σ, as so constructed, is an articulation ordering for $\mathsf{Edges}(\mathbf{H})$. □

1.5.11 Corollary – alternate characterization of articulated join plans *Let* $\mathbf{H}$ *be a hypergraph, and let* J *be a join plan for* $\mathbf{H}$*. Then* J *is articulated iff, for each node* n *of* J*, there is a join tree for* $\mathsf{Label}(n)$ *which is extendible to a join tree for all of* $\mathbf{H}$*.*

PROOF: The proof follows from the above lemma. Indeed, if J is articulated, then we may order the members of the label each node n of J to be an initial set of an articulation ordering σ for $\mathsf{Edges}(\mathbf{H})$. But then the lemma provides a join tree for the subhypergraph of $\mathbf{H}$ whose set of edges is $\mathsf{Label}(n)$, and since the articulation ordering extends to all of $\mathbf{H}$, so too does the join tree. Conversely, let J_n be a join tree for the hypergraph whose edges are $\mathsf{Label}(n)$, and assume that this join tree extends to a join tree J for all of $\mathbf{H}$. We build an articulation ordering σ for $\mathsf{Edges}(\mathbf{H})$ as follows. Pick $\sigma(1)$ arbitrarily to be any member of $\mathsf{Label}(n)$. We now proceed inductively. Given i, $1 \leq i < \mathsf{Card}(\mathsf{Edges}(\mathbf{H}))$, and having already built $\{\sigma(k) \mid 1 \leq k \leq i\}$, let $\sigma(i+1)$ be any new hyperedge H, not in the set $\{\sigma(k) \mid 1 \leq k \leq i\}$, but which is connected by an edge of J to some element $\sigma(j)$ in that set. We furthermore require that $i < \mathsf{Card}(\mathsf{Label}(n))$, then this edge must lie in $\mathsf{Label}(n)$. This is possible since J_n is a join tree, and hence itself connected. Now define $p(i+1) = j$. It is straightforward to verify that p is an articulation predecessor function

for an articulation ordering for $\mathsf{Edges}(\mathbf{H})$, and that the first $\mathsf{Card}(\mathsf{Label}(n))$ elements of this sequence lie in $\mathsf{Label}(n)$. Hence J is articulated. $\square$

2. Characterizations of Simplicity Based upon Pairwise Properties

In this section, we develop the generalizations associated with pairwise definability — that is, those identified by category PD of Figures 0.1.1 and 0.1.2. In addition to direct generalizations of (S1) and (S4) through (S7), we provide numerous other characterizations, fourteen in all. These generalizations fall into three subcategories, based upon definability, join expressions, and binary dependencies, respectively. Each category is examined in a separate subsection.

2.1 Characterizations Based upon Definability and Commutativity

In this subsection, we provide a generalization of the condition (S1) of 0.1.1 which equates pairwise and global consistency of simple relational schemata, and we identify seven other equivalent characterizations. All of these characterizations work with either the notion of "agreement in pairs is enough" or else the idea that "all congruences must commute".

We begin by recalling the key definitions within the relational context.

2.1.1 Pairwise and total consistency In the context of a finite attribute schema $\mathbf{U}$ and a simple universal relational schema $\mathbf{R}\langle\mathbf{U}\rangle$, let $(M_W)_{W\in\mathbf{U}}$ be a local $\mathbf{U}$-database over $\mathbf{R}\langle\mathbf{U}\rangle$. We say that $(M_W)_{W\in\mathbf{U}}$ is *pairwise consistent* if, for every pair $\{W_1, W_2\} \subseteq \mathbf{U}$, we have $\lambda(\Pi_{W_1}, \Pi_{W_1\cap W_2})'(M_{W_1}) = \lambda(\Pi_{W_2}, \Pi_{W_1\cap W_2})'(M_{W_2})$. $(M_W)_{W\in\mathbf{U}}$ is *globally consistent* if it is a decomposed database; *i.e.*, if it is in $\Delta\langle\mathbf{U}\rangle'(\mathsf{LDB}(\mathbf{R}\langle\mathbf{U}\rangle))$. Condition (S1) of 0.1.1 stipulates that every pairwise consistent local $\mathbf{U}$-database over $\mathbf{R}\langle\mathbf{U}\rangle$ is globally consistent iff $\mathbf{U}$ is acyclic.

In light of the discussion at the end of 1.4.2, in the context of $\mathbf{R}\langle\mathbf{U}\rangle$, both meet operations $\wedge$ and $\Cap$ correspond to intersection of attributes. Thus, to generalize the above ideas to the set-based setting, we replace intersection of the attribute sets of projections with the general operations of meet of views. Within the context of set-based schemata, there are two notions of consistency, one for each kind of meet, and so (S1) has several generalizations. Given any set S, we let $\mathcal{P}_2(S)$ denote the set of all nonempty subsets of S which contain exactly two elements. Let $\mathbf{D}$ be a set-based schema, and let X be a finite set of views of $\mathbf{D}$. Let $Y \subseteq \mathcal{P}_2(X)$. The local X-database $(M_\Gamma)_{\Gamma\in X}$ is *Y-consistent* (resp. *strongly Y-consistent*) if for each $\{\Gamma_i, \Gamma_j\} \in Y$, we have $\lambda(\Gamma_i, \Gamma_i \wedge \Gamma_j)'(M_{\Gamma_i}) = \lambda(\Gamma_j, \Gamma_i \wedge \Gamma_j)'(M_{\Gamma_j})$ (resp. $\lambda(\Gamma_i, \Gamma_i \Cap \Gamma_j)'(M_{\Gamma_i}) = \lambda(\Gamma_j, \Gamma_i \Cap \Gamma_j)'(M_{\Gamma_j})$). (Note that $\lambda(\Gamma_i, \Gamma_i \Cap \Gamma_j)'(M_{\Gamma_i}) = \lambda(\Gamma_j, \Gamma_i \Cap \Gamma_j)'(M_{\Gamma_j})$ can only hold in the case that the indicated meets are defined.) $(M_\Gamma)_{\Gamma\in X}$ is *pairwise consistent* (resp. *strongly pairwise consistent*) if it is $\mathcal{P}_2(X)$-consistent (resp. strongly $\mathcal{P}_2(X)$-consistent). It is *globally consistent* if it is a decomposed database; *i.e.*, if it is in $\Delta\langle X\rangle'(\mathsf{LDB}(\mathbf{D}))$.

We are now in a position to define the various notions of pairwise consistency implying global consistency. We say that X is *Y-definable* (resp. *strongly Y-definable*) if every Y-consistent (resp. strongly Y-consistent) local X database is globally consistent. We say that X is *all-pairs definable* (resp. *strongly all-pairs definable*) it is $\mathcal{P}_2(X)$-definable (resp. strongly $\mathcal{P}_2(X)$-definable). And finally it is *pairwise definable* (resp. *strongly pairwise definable*) if there is some $Y \subseteq \mathcal{P}_2(X)$ such that it is Y-pairwise definable (resp. strongly Y-definable). In words, the "all-pairs" conditions state that if every pair of view states is compatible, then the entire X-tuple is compatible. On the other hand, the "pairwise" conditions state that there is a certain set of pairs of views that are sufficient to check to verify total consistency. It is immediate that pairwise definability and all-pairs definability are equivalent notions. However, this equivalence of "all-pairs" and "pairwise" is not so trivial in the "strong" general case (*i.e.*, using the meet $\Cap$ from $[\mathbf{CView}(\mathbf{D})]$), because to go from pairwise definability to all-pairs definability we must establish that each pair of views has commuting congruences, which is true but not obvious (at least to the author). Indeed, we will eventually establish that these four notions are all equivalent in the general set-based context. First, however, let us verify that in the simple universal relational setting, we have indeed generalized the classical notion.

2.1.2 Observation *Let $\mathbf{R}\langle\mathbf{U}\rangle$ be a simple $\mathbf{U}$-universal relational schema. Then, any of the four conditions of all-pairs definability, strong all-pairs definability, pairwise definability, and strong pairwise definability are equivalent, and are in turn equivalent to the classical condition (S1) of pairwise consistency implying total consistency.*

PROOF: As established in 1.3.3, each pair of views in $\{\Pi_W \mid W \in \mathbf{U}\}$ has commuting congruences, and so the adjective "strong" is superfluous in the context of simple universal relational schemata. And, as we have already remarked, it is immediate that all-pairs definability and pairwise definability are equivalent even in the general context. Hence, all four notions are equivalent in this restricted context. □

We now turn to the definition of the other equivalent characterizations of pairwise definability.

2.1.3 The Chinese remainder characterization There is a famous result in the theory of commutative algebra which is known as the *Chinese Remainder Theorem* [33, Ch. 5, Thm. 17]. Roughly, it states that a system of linear congruences in a Dedekind domain has a solution iff these congruences are pairwise compatible. It has been known for some time that the theorem holds in the more general context of universal algebra, provided that certain properties of Dedekind domains are supplied as axioms [15, Ch. 5, Exer. 68]. Within our context of a set-based schema $\mathbf{D}$, the appropriate condition is formulated as follows. The set X of views is said to *satisfy the generalized Chinese remainder condition* if for any X-tuple $(M_\Gamma)_{\Gamma \in X} \in \mathsf{LDB}(\mathbf{D})^{\mathsf{Card}(X)}$ of databases of $\mathbf{R}$, if it is the case that for any $\Gamma_i, \Gamma_j \in X$ that $\mu_{\Gamma_i \wedge \Gamma_j}'(M_{\Gamma_i}) = \mu_{\Gamma_i \wedge \Gamma_j}'(M_{\Gamma_j})$, then there is an $M \in \mathsf{LDB}(\mathbf{D})$ with the property that $\mu_\Gamma'(M) = \mu_\Gamma'(M_\Gamma)$ for all $\Gamma \in X$. Note that this condition is slightly different from all-pairs definability because it works entirely with elements of $\mathsf{LDB}(\mathbf{D})$, rather than with elements

of $\prod_{\Gamma \in X} \mathsf{LDB}(\mathbf{V}_\Gamma)$. But our primary reason for introducing it is that, as we shall see, the known conditions for its solution provide an elegant means of unifying all of these consistency characterizations.

2.1.4 Pairwise and completely commuting sets of views Let X be a set of views of the set-based schema $\mathbf{D}$. We say that X is *pairwise commuting* if each pair $\{\Gamma_i, \Gamma_j\} \in \mathcal{P}_2(X)$ is commuting (see 1.3.1). Define $\mathsf{Joins}(X) = \{\bigvee Y \mid Y \subseteq X \text{ and } Y \neq \emptyset\}$. The set X is *completely commuting* if $\mathsf{Joins}(X)$ is pairwise commuting.

It is important to understand that while 1.3.3 implies that in the context of a simple universal relational schema every pair of projections is pairwise commuting, it does not imply that every subdirect decomposition into projections is completely commuting. Indeed, the equivalence of (a) and (h) in the theorem below, coupled with the equivalence of pairwise definability and condition (S1) of the classical theorem 0.1.1, show that a family of projections will be completely commuting if and only if the associated schema is acyclic. Thus, complete commutativity and pairwise commutativity must not be confused. In general, complete commutativity is a strictly stronger condition.

We are now in a position to state the main theorem identifying the equivalent consistency-based characterizations of schema decompositions.

2.1.5 Theorem — consistency characterization of schema decompositions *Let $\mathbf{D}$ be a set-based database schema, and let X be a finite set of views of $\mathbf{D}$. Then the following conditions are equivalent.*

(a) *X is pairwise definable.*

(b) *X is strongly pairwise definable.*

(c) *X is all-pairs definable.*

(d) *X is strongly all-pairs definable.*

(e) *X satisfies the generalized Chinese remainder condition.*

(f) *X is pairwise commuting and $\mathsf{Span}(X, [\mathbf{View}(\mathbf{D})])$ is a distributive sublattice of $[\mathbf{View}(\mathbf{D})]$.*

(g) *$\mathsf{Span}(X, [\mathbf{CView}(\mathbf{D})])$ is a distributive sublattice of $[\mathbf{CView}(\mathbf{D})]$.*

(h) *X is completely commuting.*

PROOF: The proof strategy is as follows. The implications (d) $\Rightarrow$ (c) $\Rightarrow$ (a), (d) $\Rightarrow$ (b) $\Rightarrow$ (a), (g) $\Rightarrow$ (f), and (g) $\Rightarrow$ (h) are immediate from the definitions, and do not require elaboration. The implications (e) $\Rightarrow$ (d), (h) $\Rightarrow$ (g), (f) $\Rightarrow$ (g), (a) $\Rightarrow$ (h), and (h) $\Rightarrow$ (e) will be established in separate propositions to follow. This provides an equivalence of all of the stated conditions. $\square$

Before embarking upon the task of proving the supporting results for the above theorem, it is appropriate to make a few observations. Each of the eight equivalent conditions of 2.1.5 fall into one of two groups. The first five, (a) through (e), are what we might call "pairwise

is sufficient" conditions. In effect, they are all variations of a generalization of (S1) of 0.1.1. Conditions (f), (g), and (h), on the other hand, do not explicitly embody any notion of "pairwise;" rather, they convey the message that "all congruences must commute." It is indeed interesting that none of the classical characterizations (S1)-(S12) mentions commutativity at all. This is perhaps not surprising though, since working directly with the congruence of a projection is not something that is often done in database theory research. Still, we shall find that this representation of pairwise definability via commutativity will prove very useful in establishing other equivalences throughout the paper.

The equivalence of (e) and (f) has been known in mathematical circles for some time. Specifically, the equivalence of these two conditions, within the slightly more general context of universal algebras, is precisely the generalization of the Chinese remainder theorem which is identified in [15, Ch. 5, Exer. 68] and in [32].

Note that there is no requirement that X be a subdirect decomposition of $\mathbf{D}$. Although the classical relational theory works only with subdirect decompositions, we have deliberately relaxed this requirement so that the above theorem may be more useful in support of other results. However, one may clearly add the hypothesis that X is a subdirect decomposition without invalidating any of the results.

We now turn to the proofs of the supporting results.

2.1.6 Proposition — ((e) ⇒ (d) of 2.1.5) *Let X be a finite set of views of the set-based schema $\mathbf{D}$. If X satisfies the generalized Chinese remainder condition, then it is also strongly all-pairs definable.*

PROOF: The proof is almost immediate. Assume that X satisfies the generalized Chinese remainder condition, and let $(M_\Gamma)_{\Gamma\in X} \in \prod_{\Gamma\in X} \mathsf{LDB}(\mathbf{V}_\Gamma)$ be all-pairs strongly compatible. For each $\Gamma \in X$, let $N_\Gamma \in {\mu_\Gamma}'^{-1}(M_\Gamma)$. Then, for all $\Gamma_i, \Gamma_j \in X$, $\lambda(\Gamma_i, \Gamma_i \Bbbwedge \Gamma_j)'(M_{\Gamma_i}) = \lambda(\Gamma_j, \Gamma_i \Bbbwedge \Gamma_j)'(M_{\Gamma_j})$, so in particular $\lambda(\Gamma_i, \Gamma_i \wedge \Gamma_j)'(M_{\Gamma_i}) = \lambda(\Gamma_j, \Gamma_i \wedge \Gamma_j)'(M_{\Gamma_j})$, and so by the generalized Chinese remainder condition, there is an $M \in \mathsf{LDB}(\mathbf{D})$ with $\mu_\Gamma'(M) = \mu_\Gamma'(N_\Gamma)$ for all $\Gamma \in X$. But clearly $M \in {\Delta\langle X\rangle}'^{-1}((M_\Gamma)_{\Gamma\in X})$, whence X is strongly all-pairs definable. □

2.1.7 Proposition — ((a) ⇒ (h) of 2.1.5) *Let X be a finite set of views of the set-based database schema $\mathbf{D}$. If X is pairwise definable, then it is also completely commuting.*

PROOF: Assume that X is pairwise definable, and let $X_1, X_2 \subseteq X$. Suppose further that $(N_1, N_2) \in \mathsf{Congr}(\bigvee X_1) \circ \mathsf{Congr}(\bigvee X_2)$. Since for all $\Gamma_i, \Gamma_j \in X_1 \cap X_2$, $\mathsf{Congr}(\bigvee X_1) \circ \mathsf{Congr}(\bigvee X_2) \subseteq \mathsf{Congr}(\Gamma_i) \circ \mathsf{Congr}(\Gamma_j) \subseteq \mathsf{Congr}(\Gamma_i) \wedge \mathsf{Congr}(\Gamma_j)$, we have that $(\mu_{\Gamma_i\wedge\Gamma_j})'(N_1) = (\mu_{\Gamma_i\wedge\Gamma_j})'(N_2)$ for all $\Gamma_i, \Gamma_j \in X_1 \cap X_2$. Hence, if we define $(M_\Gamma)_{\Gamma\in X} \in \prod_{\Gamma\in X} \mathsf{LDB}(\mathbf{V}_\Gamma)$ by

$$M_\Gamma = \begin{cases} \mu_\Gamma'(N_2) & \text{if } \Gamma_i \in X_1; \\ \mu_\Gamma'(N_1) & \text{otherwise.} \end{cases}$$

then $(M_\Gamma)_{\Gamma\in X}$ is pairwise consistent. Since X is pairwise definable, there is an $M \in \mathsf{LDB}(\mathbf{D})$ with $\Delta\langle X\rangle'(M) = (M_\Gamma)_{\Gamma\in X}$. But just by the definition of M_Γ, $(N_1, M) \in \mathsf{Congr}(\bigvee X_2)$ and

$(M, N_2) \in \mathsf{Congr}(\bigvee X_1)$, whence $(N_1, N_2) \in \mathsf{Congr}(\bigvee X_2) \circ \mathsf{Congr}(\bigvee X_1)$. Hence $\mathsf{Congr}(\bigvee X_1) \circ \mathsf{Congr}(\bigvee X_2) \subseteq \mathsf{Congr}(\bigvee X_2) \circ \mathsf{Congr}(\bigvee X_1)$. The reverse inclusion is proven similarly, whence $\{\bigvee X_1, \bigvee X_2\}$ is a commuting pair. Since X_1 and X_2 are arbitrary, it follows that X is completely commuting. □

2.1.8 Proposition — ((h) ⇒ (e) of 2.1.5) *Let X be a finite set of views of the set-based database schema $\mathbf{D}$. If X is completely commuting, then it satisfies the generalized Chinese remainder condition.*

PROOF: The proof is by induction on the number of views in X, with the case of two or fewer views constituting the basis of the induction. We begin with the basis. For X containing fewer than two views, the result is trivial. So, suppose that $X = \{\Gamma_1, \Gamma_2\}$ contains exactly two distinct views, and let $N_1, N_2 \in \mathsf{LDB}(\mathbf{D})$ be such that $(\mu_{\Gamma_1 \wedge \Gamma_2})'(N_1) = (\mu_{\Gamma_1 \wedge \Gamma_2})'(N_2)$. Since X is completely commuting, we have in particular that $\mathsf{Congr}(\Gamma_1) \circ \mathsf{Congr}(\Gamma_2) = \mathsf{Congr}(\Gamma_1) \wedge \mathsf{Congr}(\Gamma_2)$, and so $(N_1, N_2) \in \mathsf{Congr}(\Gamma_1) \circ \mathsf{Congr}(\Gamma_2)$, whence there is an $N \in \mathsf{LDB}(\mathbf{D})$ with $(N_1, N) \in \mathsf{Congr}(\Gamma_1)$ and $(N, N_2) \in \mathsf{Congr}(\Gamma_2)$. But then ${\mu_{\Gamma_1}}'(N_1) = {\mu_{\Gamma_1}}'(N)$ and ${\mu_{\Gamma_2}}'(N_2) = {\mu_{\Gamma_2}}'(N)$, so the generalized Chinese remainder condition is satisfied.

Now assume that $n > 2$ and that the result has been proven for all X with $\mathsf{Card}(X) < n$. Let $\{X_1, X_2\}$ be any partition of X with both X_1 and X_2 nonempty, and let Y be an X-indexed family $\{N_\Gamma \in \mathsf{LDB}(\mathbf{D}) \mid \Gamma \in X\}$ with the property that $(\mu_{\Gamma_i \wedge \Gamma_j})'(N_{\Gamma_i}) = (\mu_{\Gamma_i \wedge \Gamma_j})'(N_{\Gamma_j})$ for all $\Gamma_i, \Gamma_j \in Y$. Then, by the inductive hypothesis, each of the sets X_1 and X_2 satisfies the generalized Chinese remainder condition, and so there are $N_{X_1}, N_{X_2} \in \mathsf{LDB}(\mathbf{D})$ with the property that ${\mu_\Gamma}'(N_{X_1}) = {\mu_\Gamma}'(N_\Gamma)$ for all $\Gamma \in X_1$ and ${\mu_\Gamma}'(N_{X_2}) = {\mu_\Gamma}'(N_\Gamma)$ for all $\Gamma \in X_2$. By construction, we have that ${\mu_{((\bigvee X_1) \wedge (\bigvee X_2))}}'(N_{X_1}) = {\mu_{((\bigvee X_1) \wedge (\bigvee X_2))}}'(N_{X_2})$, so that we may invoke the inductive hypothesis once again to the two-element set $\{\bigvee X_1, \bigvee X_2\}$ to obtain an $N \in \mathsf{LDB}(\mathbf{D})$ with ${\mu_{(\bigvee X_1)}}'(N_{X_1}) = {\mu_{(\bigvee X_1)}}'(N)$ and ${\mu_{(\bigvee X_2)}}'(N_{X_2}) = {\mu_{(\bigvee X_2)}}'(N)$. Thus we have that ${\mu_\Gamma}'(N_\Gamma) = {\mu_\Gamma}'(N)$ for all $\Gamma \in X$, as was to be shown. □

2.1.9 Proposition — ((h) ⇒ (g) of 2.1.5) *Let X be a finite set of views of the set-based database schema $\mathbf{D}$. If X is completely commuting, then $\mathsf{Span}(X, [\mathbf{CView}(\mathbf{D})])$ is a distributive sublattice of $[\mathbf{CView}(\mathbf{D})]$.*

PROOF: Suppose that X is completely commuting, and let $\Gamma_1, \Gamma_2 \in X$. To show that $\mathsf{Span}(X, [\mathbf{CView}(\mathbf{D})])$ is a sublattice of $[\mathbf{CView}(\mathbf{D})]$, it suffices to show that each of $X \cup \{\Gamma_1 \vee \Gamma_2\}$ and $X \cup \{\Gamma_1 \wedge \Gamma_2\}$ is completely commuting, for we may build $\mathsf{Span}(X, [\mathbf{CView}(\mathbf{D})])$ by repeated application of these two constructions.

The case of $X \cup \{\Gamma_1 \vee \Gamma_2\}$ is trivial, as $\mathsf{Congr}(\Gamma_1 \vee \Gamma_2)$ commutes with the congruence of each $\Gamma \in X$ by the definition of completely commuting.

Now consider the case of $X \cup \{\Gamma_1 \wedge \Gamma_2\}$. Let $\Gamma \in X$. We know that each of $\{\Gamma_1, \Gamma_2\}$, $\{\Gamma, \Gamma_1\}$, and $\{\Gamma, \Gamma_2\}$ is a commuting pair, whence $\mathsf{Congr}(\Gamma) \circ \mathsf{Congr}(\Gamma_1 \wedge \Gamma_2) = \mathsf{Congr}(\Gamma) \circ \mathsf{Congr}(\Gamma_1) \circ \mathsf{Congr}(\Gamma_2) = \mathsf{Congr}(\Gamma_1) \circ \mathsf{Congr}(\Gamma_2) \circ \mathsf{Congr}(\Gamma) = \mathsf{Congr}(\Gamma_1 \wedge \Gamma_2) \circ \mathsf{Congr}(\Gamma)$.

To establish distributivity, let $\Gamma_1, \Gamma_2, \Gamma_3 \in \mathsf{Span}(X, [\mathbf{CView}(\mathbf{D})])$. We must show that $(\mathsf{Congr}(\Gamma_1) \circ \mathsf{Congr}(\Gamma_3)) \cap (\mathsf{Congr}(\Gamma_2) \circ \mathsf{Congr}(\Gamma_3)) = (\mathsf{Congr}(\Gamma_1) \cap \mathsf{Congr}(\Gamma_2)) \circ \mathsf{Congr}(\Gamma_3)$. Now it suffices to show that $(\mathsf{Congr}(\Gamma_1) \circ \mathsf{Congr}(\Gamma_3)) \cap (\mathsf{Congr}(\Gamma_2) \circ \mathsf{Congr}(\Gamma_3)) \subseteq (\mathsf{Congr}(\Gamma_1) \cap$

$\mathsf{Congr}(\Gamma_2)) \circ \mathsf{Congr}(\Gamma_3)$, because the reverse inclusion is trivially satisfied by any three relations. Let $(N_1, N_2) \in (\mathsf{Congr}(\Gamma_1) \circ \mathsf{Congr}(\Gamma_3)) \cap (\mathsf{Congr}(\Gamma_2) \circ \mathsf{Congr}(\Gamma_3))$. Then, since X is completely commuting, it satisfies the generalized Chinese remainder condition by the previous lemma. Define $M_1 = N_1$, $M_2 = N_1$, and $M_3 = N_2$. Then $(M_i, M_j) \in \mathsf{Congr}(\Gamma_i \wedge \Gamma_j)$ for $1 \leq i \leq 3$, so by the generalized Chinese remainder condition there is an $M \in \mathsf{LDB}(\mathbf{D})$ with $(M, M_i) \in \mathsf{Congr}(\Gamma_i)$ for $1 \leq i \leq 3$. But then $(N_1, M) \in \mathsf{Congr}(\Gamma_1) \cap \mathsf{Congr}(\Gamma_2)$ and $(M, N_2) \in \mathsf{Congr}(\Gamma_3)$, so $(N_1, N_2) \in (\mathsf{Congr}(\Gamma_1) \cap \mathsf{Congr}(\Gamma_2)) \circ \mathsf{Congr}(\Gamma_3)$, as was to be shown. □

2.1.10 Proposition — ((f) ⇒ (g) of 2.1.5) *Let X be a finite set of views of the set-based database schema $\mathbf{D}$. If X is completely commuting and $\mathsf{Span}(X, [\mathbf{View}(\mathbf{D})]$ is a distributive sublattice of $[\mathbf{View}(\mathbf{D})]$, then $\mathsf{Span}(X, [\mathbf{CView}(\mathbf{D})])$ is a distributive sublattice of $[\mathbf{CView}(\mathbf{D})]$.*

Proof: Assume that X is completely commuting and that $\mathsf{Span}(X, [\mathbf{View}(\mathbf{D})])$ is a distributive sublattice of $[\mathbf{View}(\mathbf{D})]$. Let $\Gamma \in \mathsf{Span}(X, [\mathbf{View}(\mathbf{D})])$. First of all, let $\Gamma_1, \Gamma_2, \Gamma_3 \in X$. Then, since the congruences of these three views commute pairwise, we have that $(\mathsf{Congr}(\Gamma_1) \circ \mathsf{Congr}(\Gamma_2)) \circ \mathsf{Congr}(\Gamma_3) = \mathsf{Congr}(\Gamma_2) \circ \mathsf{Congr}(\Gamma_1) \circ \mathsf{Congr}(\Gamma_3) = \mathsf{Congr}(\Gamma_2) \circ \mathsf{Congr}(\Gamma_3) \circ \mathsf{Congr}(\Gamma_1) = \mathsf{Congr}(\Gamma_3) \circ \mathsf{Congr}(\Gamma_2) \circ \mathsf{Congr}(\Gamma_1) = \mathsf{Congr}(\Gamma_3) \circ (\mathsf{Congr}(\Gamma_1) \circ \mathsf{Congr}(\Gamma_2))$, *i.e.*, $(\Gamma_1 \wedge \Gamma_2) \wedge \Gamma_3$ exists. A simple induction then establishes that the meet of any subset of X must exist in $[\mathbf{CView}(\mathbf{D})]$. Now since $[\mathbf{View}(\mathbf{D})]$ is distributive, it is easy to verify that we may represent any $\Gamma \in \mathsf{Span}(X, [\mathbf{View}(\mathbf{D})])$ in disjunctive normal form; that is, in the form $\bigvee_{i=1}^{m}(\bigwedge_{j=1}^{n} \Gamma_{ij})$, with each $\Gamma_{ij} \in X$. But by the preceding remark, $\bigvee_{i=1}^{m}(\bigwedge_{j=1}^{n} \Gamma_{ij}) = \bigvee_{i=1}^{m}(\bigwedge\!\!\!\!\bigwedge_{j=1}^{n} \Gamma_{ij})$, whence $\Gamma \in X$. Thus $\mathsf{Span}(X, [\mathbf{View}(\mathbf{D})]) = \mathsf{Span}(X, [\mathbf{CView}(\mathbf{D})])$, and each member has the same representation in disjunctive normal form. Since the operations in $[\mathbf{CView}(\mathbf{D})]$ and $[\mathbf{View}(\mathbf{D})]$ coincide when they are both defined, it follows that these two sets must be identical as sublattices, whence $\mathsf{Span}(X, [\mathbf{CView}(\mathbf{D})])$ is a distributive sublattice of $[\mathbf{CView}(\mathbf{D})])$. □

This completes the supporting propositions for 2.1.5.

2.2 Characterizations Based upon Join Expressions

As we noted in the introduction, there are three generalizations of the existence of monotone join expressions. In this subsection, we develop the one associated with pairwise definability.

2.2.1 Monotone and commuting join expressions and plans in the relational setting In the context of a simple universal relational schema $\mathbf{R}\langle\mathbf{U}\rangle$, the *join expressions* are the smallest class closed under the following operations.

(i) Each $W \in \mathbf{U}$ is a join expression.

(ii) If J_1 and J_2 are join expressions, then so too is $(J_1 \bowtie J_2)$.

Evaluation of such expressions on a local $\mathbf{U}$ database $(M_W)_{W \in \mathbf{U}}$ is in the obvious manner — we just substitute M_W for W and evaluate, with $M_{W_1} \bowtie M_{W_2}$ defined to be the $(W_1 \cup W_2)$-relation with $t \in M_{W_1} \bowtie M_{W_2}$ iff $t[W_1] \in M_{W_1}$ and $t[W_2] \in M_{W_2}$. Such an expression is called *complete* if it is an inverse to the decomposition map $\Delta\langle\mathbf{U}\rangle'$, and it is called *monotone* (or *monotonic*) if each pair of relations which is joined in the evaluation of any $(M_W)_{W \in \mathbf{U}} \in \Delta\langle\mathbf{U}\rangle'(\mathsf{LDB}(\mathbf{D}))$ is consistent, so that no tuples are discarded.

In 1.5.8, we have already defined the notion of a join plan for a hypergraph. Combining this with the natural way in which a simple universal relational schema defines a hypergraph (1.5.2), we immediately have the definition of a join plan for a simple universal relational schema. Namely, if $\mathbf{U}$ is a finite attribute schema and $\mathbf{R}$ a simple $\mathbf{U}$-universal relational schema, then a join plan for $\mathbf{R}$ is simply a join plan for $\mathcal{R}(\mathbf{U})$. There is a simple bijective correspondence between join plans and join expressions. Specifically, the join expression associated with the simple tree consisting of only one node (labelled W) is just W. For the tree with left subtree T_l and right subtree T_r of the root, the associated join expression is $(\mathcal{J}(T_l) \bowtie \mathcal{J}(T_r))$, where $\mathcal{J}(T_l)$ and $\mathcal{J}(T_r)$ are the join expressions for T_l and T_r, respectively. Such a join plan is *monotone* if the associated join expression is.

The generalization to the set-based case proceeds as follows. A *join plan* for X is a binary tree whose nodes are labelled with subsets of X, and which has the following properties.

(i) Each node has either zero or two children.

(ii) The leaves are labelled with singleton subsets of X, with each $\{\Gamma\} \subseteq X$ labelling at least one leaf.

(iii) Each nonleaf node is labelled with the union of the labels of its two children.

We call a join plan X*-complete* if the label of the root is X, and we call it just *complete* if it is X-complete and $\bigvee X = \Gamma_\top(\mathbf{D})$. It is *locally commuting* if for any two siblings n_1 and n_2 (children of a common parent) $\{\bigvee \mathsf{Label}(n_1), \bigvee \mathsf{Label}(n_2))\}$ is a commuting pair. Finally, we call a plan *sequential* if at least one child of any node in the tree is a leaf.

Because join plans in the general context correspond so closely to classical join plans in the simple universal relational setting, as well as to join plans for hypergraphs, we refer to all as simply *join plans*. However, we must be careful not to ascribe properties (such as monotonicity) to plans in a more general context, unless the underlying schema happens to be relational.

As the concept of monotonicity does not make sense in the general set-based setting, if we are to recapture it, it must be by establishing equivalence with other properties. We do this in 3.3.7, where we show that monotonicity is equivalent the property of being *articulated*. For now, we let us establish that indeed monotonicity is strictly stronger than local commutativity.

2.2.2 Proposition *Let* $\mathbf{U}$ *be a finite attribute schema, and let* $\mathbf{R}\langle\mathbf{U}\rangle$ *be a simple* $\mathbf{U}$*-universal relational schema. Then every monotone join plan on* $\mathbf{U}$ *is locally commuting, but there exist examples for which the converse is not true.*

PROOF: From the classical theory highlighted in 0.1.1, we know that the existence of such a join expression (S4) implies that pairwise consistency implies total consistency (S1), which in view of 2.1.5, implies that $\{\Pi_W \mid W \in \mathbf{U}\}$ is completely commuting. But then *every* join expression on $\mathbf{U}$ must be locally commuting.

On the other hand, consider the simple universal relational schema over $\mathbf{U} = \{AB, BC, CD\}$. Then the join plan corresponding to the expression $((AB \bowtie CD) \bowtie BC)$ is clearly not monotone, although it is locally commuting. To show that it is locally commuting, it suffices to note that this plan is is pairwise definable (condition (a) of 2.1.5 and (S1) of the classical theorem 0.1.1), and so by 2.1.5 must be fully commuting. But full commutativity of the decomposition certainly implies local commutativity of any underlying join plan. □

We are now in a position to state our main characterization regarding locally commuting join plans.

2.2.3 Theorem — equivalence of join plans and pairwise commutativity *Let* $\mathbf{D}$ *be a set-based schema, and let* X *be a finite set of views of* $\mathbf{D}$. *Then the following conditions are equivalent.*

(a) X *is pairwise definable.*

(b) X *has an* X*-complete locally commuting join plan.*

(c) X *has an* X*-complete locally commuting sequential join plan.*

PROOF: We begin by establishing that (a) ⇒ (c). First of all, assume that X is pairwise definable. Then, by 2.1.5, X is completely commuting. Hence *any* join plan for X must be locally commuting. Thus, if we let $\sigma : [1, \mathsf{Card}(X))] \to X$ be any function (with $[1, n]$ denoting the natural numbers between 1 and n inclusive), the sequential join plan depicted in Figure 1.5.1 must be locally commuting and X-complete. In this plan, n denotes the value $\mathsf{Card}(X)$.

We next establish that (b) ⇒ (a). Let T be any X-complete locally commuting join plan for X, and let $X^{(T)} = \{\bigvee \mathsf{Label}(n) \mid n \text{ is a node of } T\}$. Clearly $X^{(T)} \subseteq \mathsf{Joins}(X)$. But, by construction, $X^{(T)}$ is pairwise definable, and hence completely commuting, by 2.1.5. But then X must also be completely commuting, since $\mathsf{Joins}(X) = \mathsf{Joins}(X^{(T)})$, and hence X must be pairwise definable, again using 2.1.5.

The proof that (c) ⇒ (b) is trivial, and thus we have that the three conditions are equivalent. □

The previous theorem is a generalization of the equivalence of (S1), (S4), and (S5) of 0.1.1. But, in some sense, it is a weak one in that it the translation is not "instance-for-instance." As shown by 2.2.2, not every locally commuting join expression in the simple universal relational context is monotone. Thus, we cannot regard this result as a valid generalization at the level of individual trees, but rather only as a statement that if we have a join plan which is locally commuting, then there must also be (a possibly different) one which is monotone. The stronger result, which generalizes at the level of individual trees, is given in 3.3.7. Let us formally record the level of generalization for the current context.

2.2.4 Corollary *Let* $\mathbf{U}$ *be a finite set of attributes, and let* $\mathbf{R}\langle\mathbf{U}\rangle$ *be a simple* $\mathbf{U}$*-universal relational schema. Then, if* $\{\Pi_W \mid W \in \mathbf{U}\}$ *has a locally commuting complete join plan, it also has a monotone join plan.*

PROOF: From the above theorem, we know that if $\{\Pi_W \mid W \in \mathbf{U}\}$ has a complete locally commuting join plan, then it is pairwise definable. But from 2.1.5, this is equivalent to all-pairs definability, which is a generalization of pairwise consistency implying global consistency. Hence, from the classical result (0.1.1), we know that $\{\Pi_W \mid W \in \mathbf{U}\}$ must have a monotone join plan. □

2.3 Characterizations Based Upon Binary Dependencies

In this subsection, we generalize the classical characterizations (S6) and (S7) which are based upon the governing join dependency being equivalent to a set of multivalued dependencies (abbreviated MVD's). We start by generalizing the notion of a multivalued dependency to the set-based case.

2.3.1 Commuting binary decompositions — generalizing multivalued dependencies As we first noted in 1.2.3, for purposes of formulating properties of decompositions, the appropriate way to generalize a join dependency $\bowtie[\mathbf{U}]$ to the set-based framework is to regard it as equivalent to the set $\{\Pi_W \mid W \in \mathbf{U}\}$ of views. Since an MVD may be regarded as a join dependency with two components, the natural generalization of an MVD is a pair of views. More precisely, let $\mathbf{D}$ be a set-based schema. An *embedded commuting binary decomposition* (or *embedded CBD*) of $\mathbf{D}$ is any commuting pair Y of views. If Y is also a subdirect complementary pair, we call it a *full commuting binary decomposition* (*full CBD*), or just a *commuting binary decomposition* (*CBD*). More generally, if X is a finite set of views of $\mathbf{D}$, then an embedded CBD $Y = \{\Gamma_1, \Gamma_2\}$ of $\mathbf{D}$ is X*-complete* if $\Gamma_1 \vee \Gamma_2 = \bigvee X$. Note in particular that if X is a subdirect decomposition of $\mathbf{D}$, then the X-complete CBD's are coextensive with the full ones. Also, to avoid any possible confusion, we explicitly note that an embedded CBD may in fact be full; the embedding need not be proper.

There is a bijective correspondence between certain (full, resp. embedded) CBD's and (full, resp. embedded) multivalued dependencies. Indeed, let $\mathbf{U}$ be a finite attribute schema, and let $\psi = \bowtie[W_1, W_2]$ be a multivalued dependency (either full or embedded) over the attributes defined by $\mathbf{U}$. Then the *CBD associated with* $\bowtie[W_1, W_2]$ is just $\{\Pi_{W_1}, \Pi_{W_2}\}$, and is denoted $\mathsf{CBD}(\psi)$. If Ψ is a set of MVD's, $\mathsf{CBD}(\Psi)$ also denote the set $\{\mathsf{CBD}(\psi) \mid \psi \in \Psi\}$. In the light of 1.3.2 and 1.3.3, $\{\Pi_{W_1}, \Pi_{W_2}\}$ is a full (resp. embedded) CBD iff $\bowtie[W_1, W_2]$ is a full (resp. embedded) multivalued dependency. Thus, CBD's are a natural generalization of MVD's.

2.3.2 Equivalence of a set of CBD's to a decomposition The classical characterizations of acyclicity ((S6) and (S7) of 0.1.1) state that the governing join dependency is logically equivalent to a set of multivalued dependencies. Unfortunately, the notion of logical equivalence does not translate directly to the set-based case, because we have no notion of

legal and illegal states. However, there is an alternate way to view equivalence of constraints that generate views, such as join and multivalued dependencies. Namely, we may regard two sets of dependencies to be equivalent if they generate the same set of views. Let Θ be a set of embedded CBD's of $\mathbf{D}$, and let X continue to be a finite set of views. We say that Θ and X are *decomposition equivalent* if $\mathsf{Span}(X, [\mathbf{CView}(\mathbf{D})]) = \mathsf{Span}(\bigcup\Theta, [\mathbf{CView}(\mathbf{D})])$.

Unfortunately, decomposition equivalence, when translated to the simple universal relational context, yields something which is strictly stronger than logical equivalence. An example will help illustrate.

2.3.3 Example Let $\mathbf{U} = \{AB, BCG, CD, DE, EF\}$, and let $\mathbf{R}$ be a simple universal relational schema over $\mathbf{U}$. It is easy to see that $\mathcal{R}(\mathbf{U})$ is acyclic, and so by the classical theory, $\bowtie[\mathbf{U}]$ should be a consequence of a set of MVD's. Now it is well known that for any partition $\{\mathbf{U}_1, \mathbf{U}_2\}$ of $\mathbf{U}$, we will have $\bowtie[\mathbf{U}] \models \bowtie[\overline{\mathbf{U_1}}, \overline{\mathbf{U_2}}]$ [24, Sec. 4]. (Recall that $\overline{\mathbf{U}}$ denotes $\bigcup\mathbf{U}$.) However, consider the specific choices of $\mathbf{U}_1 = \{AB, CD\}$ and $\mathbf{U}_2 = \{BCG, DE, EF\}$. Then the resulting MVD is $\bowtie[ABCD, BCDEFG]$, and neither Π_{ABCD} nor Π_{BCDEFG} is in $\mathsf{Span}(\{\Pi_W \mid W \in \mathbf{U}\}, [\mathbf{CView}(\mathbf{R})])$. The problem is that neither the AB and CD projections nor the BCG and DEF projections join completely, and so neither Π_{ABCD} nor Π_{BCDEFG} are constructible by join and meet operations on elements of $\{\Pi_W \mid W \in \mathbf{U}\}$. To remedy this situation and obtain coalescence of logical and decomposition equivalence, we must be more selective about which MVD's we allow.

2.3.4 The fundamental implicants of a join dependency Let $\mathbf{U}$ be a finite attribute schema, and let $\psi = \bowtie[W_1, W_2]$ be a total MVD on the attribute set $\cup\mathbf{U}$. Call ψ a *fundamental implicant* of $\bowtie[\mathbf{U}]$ if there is a partition $\{\mathbf{U}_1, \mathbf{U}_2\}$ of $\mathbf{U}$ such that we have the following three properties

(fi-i) For $i = 1, 2$, $W_i = \overline{\mathbf{U}_i}$;

(fi-ii) For $i = 1, 2$, $\Pi_{W_i} = \bigvee\{\Pi_W \mid W \in \mathbf{U}_i\}$; (*i.e.*, the defining components *join completely*).

(fi-iii) For $i = 1, 2$, there are $U_i \in \mathbf{U}_i$ such that $\{U_1, U_2\}$ is an articulation pair for $\mathcal{R}(\mathbf{U})$.

The set of all fundamental implicants of $\bowtie[\mathbf{U}]$ is denoted $\mathsf{FundImpl}(\mathbf{U})$, and the set of all projections of the form Π_W for some $\bowtie[W, V] \in \mathsf{FundImpl}(\mathbf{U})$ is called the set of all *fundamental projections* of $\mathbf{U}$, and is denoted by $\mathsf{FundProj}(\mathbf{U})$.

The basic idea behind fundamental implicants is to avoid sets of attributes that cannot arise as a join (as opposed to a union) of members of $\{\Pi_W \mid W \in \mathbf{U}\}$. Note that, in the above example, $\bowtie[ABCD, BCDEFG]$ is not a fundamental implicant of $\bowtie[\mathbf{U}]$. We next establish that whenever a join dependency is a logical consequence of a set of MVD's, then it is a logical consequence of some of its fundamental implicants.

It is well-known that any MVD satisfying (fi-i) above is a consequence of $\bowtie[\mathbf{U}]$, and that if $\bowtie[\mathbf{U}]$ is the consequence of any set of MVD's, then it is the consequence of those MVD's satisfying (fi-i) [24, Sec. 4]. We will show that, after enforcing conditions (fi-ii) and (fi-iii)

as well, the resulting set of MVD's is still robust enough to entail $\bowtie[\mathbf{U}]$. Furthermore, upon restricting our attention to fundamental implicants, we will obtain the desired equivalence of logical and decomposition equivalence.

2.3.5 Lemma *Let* $\mathbf{U}$ *be a finite attribute schema, and let* $\{\mathbf{U}_1, \mathbf{U}_2\}$ *be a partition of* $\mathbf{U}$. *Then* $\bowtie[\overline{\mathbf{U}_1}, \overline{\mathbf{U}_2}]$ *is a fundamental implicant of* $\bowtie[\mathbf{U}]$ *iff* $\mathbf{U}$ *has a join graph* G *with an edge* e *such that removing* e *from* G *results in two disconnected subgraphs* G_1 *and* G_2, *whose sets of vertices are* $\mathbf{U}_1$ *and* $\mathbf{U}_2$, *respectively.*

PROOF: In this proof, we make use of the well-known (and obvious) fact that to compute the join of a local $\mathbf{U}$-database, it suffices to compute each of the joins indicated by the edges in a join graph (in any order).

First suppose that $\bowtie[\overline{\mathbf{U}_1}, \overline{\mathbf{U}_2}]$ is a fundamental implicant of $\bowtie[\mathbf{U}]$. Let G_1 and G_2 be join graphs for $\mathbf{U}_1$ and $\mathbf{U}_2$, respectively, and let $U_1 \in \mathbf{U}_1$ and $U_2 \in \mathbf{U}_2$ be such that $\{U_1, U_2\}$ is an articulation pair for $\mathcal{R}(\mathbf{U})$. (Since the elements of $\mathbf{U}_1$ and of $\mathbf{U}_2$ each join completely, G_1 and G_2 exist by the above remark.) Let G be the graph obtained by connecting G_1 and G_2 via an edge labelled $U_1 \cap U_2$ from U_1 to U_2. Then G is a join graph for $\mathbf{U}$. Indeed, we must have that $U_1 \cap U_2 = (\overline{\mathbf{U}_1}) \cap (\overline{\mathbf{U}_2})$, else to complete $G_1 \cup G_2$ to a join graph we would have to add another edge, say from $V_1 \in \mathbf{U}_1$ to $V_2 \in \mathbf{U}_2$, with $V_1 \cap V_2 \not\subseteq U_1 \cap U_2$. However, the necessity of adding such an edge would contradict that $\{U_1, U_2\}$ is an articulation pair. Thus, in particular, if $V_1 \in \mathbf{U}_1$ and $V_2 \in \mathbf{U}_2$, then $V_1 \cap V_2 \subseteq (\overline{\mathbf{U}_1}) \cap (\overline{\mathbf{U}_2})$, so that we may identify a $(V_1 \cap V_2)$-path from V_1 to V_2 by following first a $(V_1 \cap U_1)$-path from V_1 to U_1, then the edge from U_1 to U_2, and finally a $(U_2 \cap V_2)$-path from U_2 to V_2. (Remember that G_1 and G_2 are join graphs for their nodes.) Hence G is a join graph for $\mathbf{U}$.

Conversely, assume that G is a join graph for $\mathbf{U}$ with the property that removing the single edge e results in two disconnected subgraphs G_1 and G_2, whose vertices are $\mathbf{U}_1$ and $\mathbf{U}_2$, respectively. Then $\{U_1, U_2\}$ must be an articulation pair for $\mathbf{U}$. Furthermore, in light of the remark made at the beginning of this proof, the subtrees G_1 and G_2 must each join completely. Hence $\bowtie[\overline{\mathbf{U}_1}, \overline{\mathbf{U}_2}]$ is a fundamental implicant of $\bowtie[\mathbf{U}]$, as required. □

2.3.6 Proposition *Let* $\mathbf{U}$ *be a finite attribute schema. Then, if there is a set* Ψ *of MVD's such that* $\Psi \models \bowtie[\mathbf{U}]$, *then also* $\mathsf{FundImpl}(\mathbf{U}) \models \bowtie[\mathbf{U}]$. *Furthermore, in this case, for each* $W \in \mathbf{U}$ *there is a set* $Y \subseteq \mathsf{FundProj}(\mathbf{U})$ *such that* $W = \bigcap\{V \mid \Pi_V \in Y\}$.

PROOF: Since we know that there is a set Ψ of MVD's such that $\Psi \models \bowtie[\mathbf{U}]$, by the classical theorem (0.1.1), $\mathbf{U}$ is acyclic. Hence, by 0.1.1 (or 1.5.7), $\mathbf{U}$ has a join tree T. First of all, note that given any tree T whatever, there must be at least one node ν of T which has incidence one; *i.e.*, there is exactly one edge connected to ν. Thus, for any tree T, we may find a sequence $(\nu_1, T_1), (\nu_2, T_2), \ldots (\nu_{n-1}, T_{n-1})$ in which each $T_0 = T$, each T_i is a subtree of T, each ν_i is a node of T_{i-1} of incidence one, and for each i, we have that $\mathsf{Nodes}(T_i) = \mathsf{Nodes}(T_{i-1}) \setminus \{\nu_i\}$. Here n denotes the number of nodes of T. The key to our proof is to apply this construction to a join tree T of $\mathbf{U}$. In so doing, we obtain a sequence of embedded MVD's $\psi_i = [\nu_i, \bigcup \mathsf{Nodes}(T_i)]$. Since the nodes used to define each such MVD lie within a subtree of T (remember that a tree is connected, by definition) the projections

defined by the collection of nodes of each such subtree must join completely, and so each ψ_i is a consequence of $\bowtie[\mathbf{U}]$. Now, by using ψ_{i+1} to decompose the $\bigcup \mathsf{Nodes}(T_i)$ component of ψ_i, we may easily conclude that $\{\psi_i \mid 1 \leq i \leq \mathsf{Card}(\mathbf{U}) - 1\} \models \bowtie[\mathbf{U}]$.

It remains to show that we may obtain a set of fundamental implicants of $\bowtie[\mathbf{U}]$ from the ψ_i's. To do so, for each i, let e_i be the edge which connects ν_i to T_i. Since T is a tree, any edge meets the requirements of 2.3.5. Therefore, in particular, we may remove e_i from T and obtain a fundamental implicant φ_i of $\bowtie[\mathbf{U}]$, as explained in 2.3.5. But it is clear that $\varphi_i \models \psi_i$, since the pair of trees defining ψ_i is obtained from the pair defining φ_i by deleting nodes which are distinct from the pair which connect the two subtrees, yet which preserve connectivity. Thus, $\{\varphi_i \mid 1 \leq i \leq \mathsf{Card}(\mathbf{U})\}$ is a set of fundamental implicants of $\bowtie[\mathbf{U}]$ which logically implies $\bowtie[\mathbf{U}]$, as required. The existence of the set Y of fundamental projections follows directly from this construction. □

We are now in a position to establish that the notions of logical equivalence and decomposition equivalence coincide for fundamental implicants.

2.3.7 Proposition *Let* $\mathbf{U}$ *be a finite attribute schema, and let* $\mathbf{R}$ *be a simple* $\mathbf{U}$*-universal relational schema. Then* $\mathsf{CBD}(\mathsf{FundImpl}(\mathbf{U}))$ *and* $\{\Pi_W \mid W \in \mathbf{U}\}$ *are decomposition equivalent iff* $\mathsf{FundImpl}(\mathbf{U})$ *and* $\bowtie[\mathbf{U}]$ *are logically equivalent.*

PROOF: We have designed the notion of fundamental implicant to ensure that we must always have that $\mathsf{Span}(\mathsf{FundProj}(\mathbf{U}), [\mathbf{CView}(\mathbf{R})]) \subseteq \mathsf{Span}(\{\Pi_W \mid W \in \mathbf{U}\}, [\mathbf{CView}(\mathbf{R})])$. Thus, we only need to establish the reverse direction. First, suppose that $\mathsf{CBD}(\mathsf{FundImpl}(\mathbf{U}))$ and $\{\Pi_W \mid W \in \mathbf{U}\}$ are decomposition equivalent. Since these two sets of dependencies define the same partial sublattice, by 1.4.4 $\{\Pi_W \mid W \in \mathbf{U}\}$ must be a subdirect decomposition of $(R, \{\bowtie[\mathbf{U}]\})$ in each context, where $(R, \{\bowtie[\mathbf{U}]\})$ is the schema $\mathbf{R}$ augmented with the join dependency $\bowtie[\mathbf{U}]$. In particular, $\{\Pi_W \mid W \in \mathbf{U}\}$ is a subdirect decomposition of $(R, \{\bowtie[\mathbf{U}]\})$ in the context of $\mathsf{CBD}(\mathsf{FundImpl}(\mathbf{U}))$. But this means that these MVD's logically imply $\bowtie[\mathbf{U}]$, as required.

Conversely, suppose that $\mathsf{FundImpl}(\mathbf{U})$ and $\bowtie[\mathbf{U}]$ are logically equivalent. Then, by the previous lemma, we know that for each $W \in \mathbf{U}$ there is a subset $Y \subseteq \mathsf{FundProj}(\mathbf{U})$ such that $W = \bigcap\{V \mid \Pi_V \in Y\}$. But this means that $\Pi_W = \bigwedge Y$. Hence, all elements of $\{\Pi_W \mid W \in \mathbf{U}\}$ are also in $\mathsf{Span}(\mathsf{FundProj}(\mathbf{U}), [\mathbf{CView}(\mathbf{R})])$, and so we must have that $\mathsf{Span}(\mathsf{FundProj}(\mathbf{U}), [\mathbf{CView}(\mathbf{R})]) = \mathsf{Span}(\{\Pi_W \mid W \in \mathbf{U}\}, [\mathbf{CView}(\mathbf{R})])$. □

We now turn to the generalization of the notion of conflict freedom. In the classical theory, a set Ψ of MVD's is conflict free if it has two properties. The first is that it *does not split keys*. The second may be taken from any number of equivalent properties, the most basic of which is that Ψ is the consequence of a single join dependency, which we term *JD generated*. We begin with a description of our generalization of these two properties.

2.3.8 Key splitting and its generalization — meet splitting Let $\mathbf{U}$ be a finite attribute schema, and let Ψ be a set of MVD's. The *key* of the MVD $\bowtie[W_1, W_2] \in \Psi$ is $W_1 \cap W_2$. It is said that Ψ *does not split keys* if for any W which is the key of some MVD in Ψ and for any other $\bowtie[V_1, V_2] \in \Psi$, either $W \subseteq V_1$ or $W \subseteq V_2$.

To generalize this notion to the set-based context, let $\mathbf{D}$ be a set-based schema, and let Θ be a set of CBD's of $\mathbf{D}$. We say that Θ *does not split meets* if for any $\{\Gamma_{11}, \Gamma_{12}\}, \{\Gamma_{21}, \Gamma_{22}\} \in \Theta$ we have that either $\Gamma_{11} \mathbb{\wedge} \Gamma_{12} \leq \Gamma_{21}$ or else $\Gamma_{11} \mathbb{\wedge} \Gamma_{12} \leq \Gamma_{22}$.

2.3.9 Lemma *Let* $\mathbf{U}$ *be a finite attribute schema, and let* Ψ *be a set of full MVD's over* $\overline{\mathbf{U}}$. *Then* Ψ *does not split keys iff* $\mathsf{CBD}(\Psi)$ *does not split meets.*

PROOF: Since meet translates to column intersection (1.3.3), the proof is immediate. □

2.3.10 Lemma *Let* $\mathbf{U}$ *be a finite attribute schema. Then* $\mathsf{FundImpl}(\mathbf{U})$ *does not split meets.*

PROOF: Let $\bowtie[W_{11}, W_{12}]$ and $\bowtie[W_{21}, W_{22}]$ each be fundamental implicants of $\bowtie[\mathbf{U}]$. In view of 1.3.3, it suffices to establish that either $W_{11} \cap W_{22} \subseteq W_{21}$ or else $W_{11} \cap W_{22} \subseteq W_{22}$. We know (2.3.5) that there is a join tree T for $\mathbf{U}$ with a distinguished edge e such that if we delete e from T, then the two remaining subtrees T_1 and T_2 have the property that $W_{21} = \bigcup \mathsf{Nodes}(T_1)$ and $W_{22} = \bigcup \mathsf{Nodes}(T_2)$. We also know that there are attributes sets $U_1, U_2 \in \mathbf{U}$ such that $W_{11} \cap W_{12} = U_1 \cap U_2$. If U_1 and U_2 are in the same subtree of T (either T_1 or T_2), then the result is trivial. Otherwise, there must be a path in T from U_1 to U_2 which includes the edge e. Now letting W denote the label of e, we know that $U_1 \cap U_2 \subseteq W$. However, $W \subseteq W_{21} \cap W_{22}$, which establishes the desired result. □

2.3.11 JD-generated and conflict-free sets of MVD's Let U be any finite set of attributes. A set Ψ of full MVD's whose attributes are precisely U is *JD-generated* if there is some finite attribute schema $\mathbf{U}$ with $U = \overline{\mathbf{U}}$ such that whenever ψ is an MVD on attributes U with the property that $\bowtie[\mathbf{U}] \models \psi$ iff $\Psi \models \psi$.

We *define* a set Ψ of MVD's to be *conflict free* if it does not split keys and it is JD-generated. (See the comments following Definition A of [4, Sec. 4] for a justification of this formulation — it is indeed equivalent to the more traditional representation, as may be found in [23, Def. 13.20]).

2.3.12 Generalizing conflict freedom to the set-based context Now let $\mathbf{D}$ be a set-based schema. A set Θ of CBD's of $\mathbf{D}$ is called *decomposition generated* if there is a finite set X of pairwise-definable views of $\mathbf{D}$ such that whenever θ is a CBD with the property that $\mathsf{Span}(\theta, [\mathbf{CView}(\mathbf{D})]) \subseteq \mathsf{Span}(X, [\mathbf{CView}(\mathbf{D})])$ iff $\mathsf{Span}(\theta, [\mathbf{CView}(\mathbf{D})]) \subseteq \mathsf{Span}(\cup\Theta, [\mathbf{CView}(\mathbf{D})])$. Θ is *decomposition-conflict free* if it does not split meets and is decomposition generated. If Ψ is a set of MVD's, then we say that Ψ is *decomposition conflict free* if $\mathsf{CBD}(\Psi)$ is decomposition conflict free.

2.3.13 Lemma *Let* $\mathbf{U}$ *be a finite attribute schema such that* $\mathcal{R}(\mathbf{U})$ *is acyclic. Then,* $\mathsf{FundImpl}(\mathbf{U})$ *is conflict free and* $\mathsf{CBD}(\mathsf{FundImpl}(\mathbf{U}))$ *is decomposition conflict free.*

PROOF: It is immediate that $\mathsf{FundImpl}(\mathbf{U})$ does not split keys, and by 2.3.9, it follows that $\mathsf{CBD}(\mathsf{FundImpl}(\mathbf{U}))$ does not split meets. From the classical theory, since $\bowtie[\mathbf{U}]$ is acyclic,

we know that there is a set Ψ of MVD's such that $\Psi \models \bowtie[\mathbf{U}]$. But then, by 2.3.6 and 2.3.7, we can conclude that $\mathsf{FundImpl}(\mathbf{U})$ is such as set of MVD's, and that it is both logically and decomposition equivalent to $\bowtie[\mathbf{U}]$. Thus, in particular, $\mathsf{FundImpl}(\mathbf{U})$ is JD-generated and $\mathsf{CBD}(\mathsf{FundImpl}(\mathbf{U}))$ is decomposition generated. Hence $\mathsf{FundImpl}(\mathbf{U})$ is conflict free and $\mathsf{CBD}(\mathsf{FundImpl}(\mathbf{U}))$ is decomposition conflict free. □

It is prudent at this point to present a cautionary example, to emphasize that the equivalence of JD-generation and decomposition generation only holds in the context of an acyclic join dependency. Indeed, a cyclic join dependency need have no fundamental implicants at all, as illustrated by the following example.

2.3.14 Example Let $\mathbf{U} = \{AB, BC, CD, DE, EF, FA\}$. Then it is not difficult to see that $\bowtie[\mathbf{U}]$ has no fundamental implicants. (Just look at any join graph.) Nonetheless, if we set $\mathbf{U}_1 = \{AB, DE, FA\}$ and $\mathbf{U}_2 = \{BC, CD, EF\}$, then $\bowtie[\mathbf{U}] \models \bowtie[ABDEF, BCDEF] = \bowtie[\overline{\mathbf{U}_1}, \overline{\mathbf{U}_2}]$, which is a nontrivial MVD. Hence there are MVD's which are implied by $\bowtie[\mathbf{U}]$ but which are not implied by the fundamental implicants of $\bowtie[\mathbf{U}]$, and so the fundamental implicants of $\bowtie[\mathbf{U}]$ do not form an MVD basis.

Finally, we can state and prove the connection between the MVD- and CBD-framework concepts

2.3.15 Proposition — the use of fundamental implicants and decomposition in the simple universal relational characterization of schema properties *Let* $\mathbf{U}$ *be a finite attribute schema, and let* $\mathbf{R}$ *be a simple* $\mathbf{U}$*-universal relational schema over* $\mathbf{U}$*. Then the following conditions are equivalent.*

(a) $\mathcal{R}(\mathbf{U})$ *is acyclic.*

(b) $\bowtie[\mathbf{U}]$ *is logically equivalent to a set of MVD's.*

(c) $\bowtie[\mathbf{U}]$ *is logically equivalent to a set of its fundamental implicants.*

(d) $\bowtie[\mathbf{U}]$ *is decomposition equivalent to a set of its fundamental implicants.*

(e) $\bowtie[\mathbf{U}]$ *is logically equivalent to a set of conflict-free MVD's.*

(f) $\bowtie[\mathbf{U}]$ *is logically equivalent to a conflict-free set of its fundamental implicants.*

(g) $\bowtie[\mathbf{U}]$ *is decomposition equivalent to a decomposition conflict free set of its fundamental implicants.*

PROOF: The equivalence of (a), (b), and (e) is just part of the classical result 0.1.1. (b) $\Rightarrow$ (c) follows from 2.3.6, and (c) $\Rightarrow$ (b) is trivial. (c) $\iff$ (d) follows from 2.3.7. (f) $\iff$ (g) follows from 2.3.7 and 2.3.13. (d) $\Rightarrow$ (g) follows since since any set of fundamental implicants is decomposition generated just by definition, and does not split meets by 2.3.10. Finally, (e) $\Rightarrow$ (c) is true since any set of fundamental implicants is in particular a set of MVD's. (g) $\Rightarrow$ (d) is trivial. We thus have equivalence of all of these items. □

The upshot of all this is that, in the classical theory, and in the context of any acyclic schema, we may replace "logically equivalent" with "decomposition equivalent" and "conflict free" with "decomposition conflict free," as long as we also replace "MVD" with "fundamental implicant". This implies that our main theorem, stated below, is a bona-fide generalization of the classical relational case.

2.3.16 Theorem — characterization of decomposition properties in terms of CBD's *Let* $\mathbf{D}$ *be a set-based schema, and let* X *be a finite set of views of* $\mathbf{D}$*. Then the following conditions are equivalent.*

(a) X *is pairwise definable.*

(b) X *is decomposition equivalent to a set of embedded CBD's.*

(c) X *is decomposition equivalent to a set of conflict-free embedded CBD's.*

(d) X *is decomposition equivalent to a set of* X*-complete CBD's.*

(e) X *is decomposition equivalent to a set of conflict-free* X*-complete CBD's.*

PROOF: The implications (e) $\Rightarrow$ (c) $\Rightarrow$ (b) and (e) $\Rightarrow$ (d) $\Rightarrow$ (b) are trivial. We establish that (b) $\Rightarrow$ (a) and that (a) $\Rightarrow$ (e) via two propositions, given below. $\square$

Notice that our theorem talks about embedded CBD's, and not just full ones. We do this even though we have not detailed the connection to embedded MVD's, which is straightforward but extremely technical. Of course, the embedded CBD statements ((b) and (c)) may be deleted, and the remaining three characterizations still are equivalent.

We now proceed with the proofs of the supporting statements for this theorem.

2.3.17 Proposition — ((b) $\Rightarrow$ (a) of 2.3.16) *Let* $\mathbf{D}$ *be a set-based schema, and let* X *be a finite set of views of* $\mathbf{D}$*. If* X *is decomposition equivalent to a set of embedded CBD's, then* X *is pairwise definable.*

PROOF: Let Θ be a set of embedded CBD's to which X is equivalent. Then $\mathsf{Span}(X, [\mathbf{CView}(\mathbf{D})]) = \mathsf{Span}(\bigcup \Theta, [\mathbf{CView}(\mathbf{D})])$ is strongly Θ-definable (see 2.1.1). Hence, by 2.1.5, $\mathsf{Span}(X, [\mathbf{CView}(\mathbf{D})])$ is completely commuting. But then X is also completely commuting, and hence pairwise definable, again by 2.1.5. $\square$

2.3.18 The extremal CBD's generated by a set of views Let $\mathbf{D}$ be a set-based schema, and X a finite set of views of $\mathbf{D}$. Define

$$\mathsf{ExtrCBD}(X) = \{\{\Gamma, \bigvee(X \setminus \{\Gamma\})\} \mid \Gamma \in X \text{ and } \{\Gamma, \bigvee(X \setminus \{\Gamma\})\} \text{ is a commuting pair}\}.$$

It is immediate that $\mathsf{ExtrCBD}(X)$ is a set of X-complete CBD's, which we call the *extremal CBD's* of X. We have no guarantee in general that $\mathsf{ExtrCBD}(X)$ is nonempty, but we fortunately have the following result.

2.3.19 Lemma *Let* $\mathbf{D}$ *be a set-based schema, and let* X *be a finite set of views. Assume further that* X *is pairwise definable. Then, for each* $\Gamma \in X$*, the pair* $\{\Gamma, \bigvee(X \setminus \{\Gamma\})\}$ *is in* $\mathsf{ExtrCBD}(X)$*, and* $\mathsf{ExtrCBD}(X)$ *and* X *are decomposition equivalent. Furthermore,* $\mathsf{ExtrCBD}(X)$ *is conflict free.*

PROOF: Let $\Gamma_1, \Gamma_2 \in X$. Since we are taking X to be pairwise definable, by 2.1.5 it is completely commuting, and so each of $\{\Gamma_1, \bigvee(X \setminus \{\Gamma_1\})\}$ and $\{\Gamma_2, \bigvee(X \setminus \{\Gamma_2\})\}$ is in $\mathsf{ExtrCBD}(X)$. Since each $\Gamma \in X$ occurs in as one of the components of a pair in $\mathsf{ExtrCBD}(X)$, it follows that these two sets are decomposition equivalent.

Now since X is pairwise definable, it is the case that $\mathsf{Span}(X, [\mathbf{CView}(\mathbf{D})])$ is distributive, by 2.1.5. Thus, $\Gamma_1 \mathbin{\mathbb{\wedge}} (\bigvee(X \setminus \{\Gamma_1\})) = \bigvee\{\Gamma_1 \mathbin{\mathbb{\wedge}} \Omega \mid \Omega \in X \setminus \{\Gamma_1\}\}$. But for each $\Gamma \in X \setminus \{\Gamma_1\}$, either $\Gamma = \Gamma_2$, or else $\Gamma_2 \in X \setminus \{\Gamma_2\}$. Thus, for each such Γ we have $\Gamma_1 \mathbin{\mathbb{\wedge}} \Gamma \leq \bigvee_{\Omega \in X \setminus \{\Gamma_2\}} \Omega$, and so $\Gamma_1 \mathbin{\mathbb{\wedge}} (\bigvee(X \setminus \{\Gamma_1\}) \leq \bigvee(X \setminus \{\Gamma_2\})$. Hence $\mathsf{ExtrCBD}(X)$ does not split meets.

Finally, it is immediate that $\mathsf{ExtrCBD}(X)$ is decomposition generated by the set X, whence it is conflict free. □

2.3.20 Proposition — ((a) ⇒ (e) of 2.3.16) *Let* $\mathbf{D}$ *be a set-based schema, and let* X *be a finite set of views of* $\mathbf{D}$*. Then, if* X *is pairwise definable, it is decomposition equivalent to a set of conflict-free* X*-complete CBD's.*

PROOF: The appropriate set of CBD's is $\mathsf{ExtrCBD}(X)$. By the previous two lemmata, we have that $\mathsf{ExtrCBD}(X)$ is conflict free and decomposition equivalent to X. □

3. Characterizations of Simplicity Involving Acyclicity

We now turn to the characterizations of simplicity identified by the categories ACY and PDA of Figures 0.1.1 and 0.1.2; that is, those characterizations which involve hypergraph acyclicity.

3.1 Translation of Hypergraph Properties to the Database Context

We first examine those characterizations identified by the category ACY; *i.e.*, acyclicity alone. In 1.5, we have already identified the essential features of hypergraphs and the equivalent characterizations of acyclicity without recourse to specific database properties. However, as we noted in 1.5.2, a simple universal relational schema *is*, for all practical purposes, a hypergraph. On the other hand, upon moving to the general set-based context, it is not so clear that we can identify an associated hypergraph, since arbitrary set-based schemata do not have attributes. However, by once again regarding the fundamental building blocks as views rather than attributes, we can provide a satisfactory definition.

3.1.1 Hypergraphs of schemata Let $\mathbf{D}$ be a set-based schema, and let X be a finite set of views of $\mathbf{D}$. Define $\langle\Gamma\rangle_X = \{\bigwedge Y \mid Y \subseteq X \text{ and } Y \neq \emptyset \text{ and } \bigwedge Y \leq \Gamma\}$. We call $\langle\Gamma\rangle_X$ the *meet ideal generated by* Γ *in* $[\mathbf{CView}(\mathbf{D})]$. (Note that $\bigwedge Y$ need not exist for each $Y \subseteq X$; implicit in the notation is that we only include those which do exist.) The hypergraph that we associate with X has exactly the meet ideals of the elements of X as its nodes. More precisely, we define the *hypergraph of* X, denoted $\mathcal{H}(X)$, as follows.

(i) $\mathsf{Nodes}(\mathcal{H}(X)) = \{\bigwedge Y \mid Y \subseteq X \text{ and } Y \neq \emptyset\}$

(ii) $\mathsf{Edges}(\mathcal{H}(X)) = \{\langle\Gamma\rangle_X \mid \Gamma \in X\}$.

Now if $\mathbf{U}$ is a finite attribute schema, then we may define a natural correspondence between the edges of $\mathcal{H}(\mathbf{R}\langle\mathbf{U}\rangle)$ and of $\mathcal{R}(\mathbf{U})$. Specifically, we define $\mathcal{F}_U : \mathsf{Edges}(\mathcal{R}(\mathbf{U})) \to \mathsf{Edges}(\mathcal{H}(\Pi\mathsf{Views}(\mathbf{U})))$ by $W \mapsto \langle\Gamma_W\rangle_{\Pi\mathsf{Views}(\mathbf{U})}$. (Recall from 1.2.3 that $\Pi\mathsf{Views}(\mathbf{U})$ denotes the set $\{\Pi_W \mid W \in \mathbf{U}\}$ of views.) Clearly $\mathcal{F}_U$ is a bijection. We will further show that it preserves and reflects many important properties. However, let us first illustrate the relationship between $\mathcal{R}(\mathbf{U})$ and $\mathcal{H}(\mathbf{R}\langle\mathbf{U}\rangle)$ with a simple example.

3.1.2 Example Let $\mathbf{U} = \{ABC, CDE, EFA, ACE\}$. (This schema is also used as an example in [4], so the reader may look there for an elaboration of more classical properties.) The corresponding set of views $\Pi\mathsf{Views}(\mathbf{U})$ is $X = \{\Pi_{ABC}, \Pi_{CDE}, \Pi_{EFA}, \Pi_{ACE}\}$. In light of 1.3.3, all appropriate congruences commute, so we have

$$\langle\Pi_{ABC}\rangle_X = \{\Pi_{ABC}, \Pi_{AC}, \Pi_A, \Pi_C, \Pi_\emptyset\}$$

$$\langle\Pi_{CDE}\rangle_X = \{\Pi_{CDE}, \Pi_{CE}, \Pi_C, \Pi_E, \Pi_\emptyset\}$$

$$\langle\Pi_{EFA}\rangle_X = \{\Pi_{EFA}, \Pi_{EA}, \Pi_A, \Pi_E, \Pi_\emptyset\}$$

$$\langle\Pi_{ACE}\rangle_X = \{\Pi_{ACE}, \Pi_{EA}, \Pi_{AC}, \Pi_{CE}, \Pi_A, \Pi_C, \Pi_E, \Pi_\emptyset\}$$

Let us look more closely at how $\langle\Pi_{ABC}\rangle_X$ is obtained. According to the above definition, we must find the meet of each nonempty subset of X which consists of commuting views, and retain those subsets whose meets are smaller than Π_{ABC}. Now, as noted above, all views in this example have commuting congruences, so we note that $\Pi_{ABC} \wedge \Pi_{ACE} = \Pi_{AC}$, $\Pi_{ABC} \wedge \Pi_{EFA} = \Pi_A$, $\Pi_{ABC} \wedge \Pi_{CDE} = \Pi_C$, and $\Pi_{ABC} \wedge \Pi_{CDE} \wedge \Pi_{EFA} = \Pi_\emptyset$, so $\{\Pi_{ABC}, \Pi_{AC}, \Pi_A, \Pi_C, \Pi_\emptyset\} \subseteq \langle\Pi_{ABC}\rangle_X$. On the other hand, further computation shows that no other views can be obtained in this fashion, so that in fact $\{\Pi_{ABC}, \Pi_{AC}, \Pi_A, \Pi_C, \Pi_\emptyset\} = \langle\Pi_{ABC}\rangle_X$. The other three sets are obtained similarly. In light of definition (ii) in 3.1.1 above, these four sets are precisely the hyperedges of $\mathcal{H}(X)$, and in light of (i) of 3.1.1 the set of nodes of $\mathcal{H}(X)$ is the union of these four sets.

The key property which we seek to preserve in the generalization is acyclicity. This is indeed the case, as implied by the following.

3.1.3 Proposition *Let* $\mathbf{U}$ *be any finite attribute schema. Then* $\mathcal{R}(\mathbf{U})$ *is closed cyclic iff* $\mathcal{H}(\Pi\mathsf{Views}(\mathbf{U}))$ *is.*

PROOF: First of all, assume that $\mathcal{R}(\mathbf{U})$ is cyclic, and let $W \subseteq \overline{\mathbf{U}}$ be a closed block of this hypergraph. Let $(E_1, E_2) \in \mathsf{Edges}(\mathsf{HGen}(\mathcal{R}(\mathbf{U}), W)) \times \mathsf{Edges}(\mathsf{HGen}(\mathcal{R}(\mathbf{U}), W))$ be an articulation pair for $\mathsf{HGen}(\mathcal{R}(\mathbf{U}), N)$, where $N = \bigcup\{\langle \Pi_V \rangle_{\Pi\mathsf{Views}(\mathbf{U})} \mid V \in \mathbf{U} \text{ and } V \subseteq W\}$. Then it is not difficult to see that $\langle \Pi_{E_1 \cap E_2} \rangle_{\Pi\mathsf{Views}(\mathbf{U})}$ is an articulation pair for $\mathsf{HGen}(\mathcal{H}(\Pi\mathsf{Views}(\mathbf{U})), N)$. Indeed, if F_1 and F_2 are hyperedges of $\mathsf{HGen}(\mathcal{R}(\mathbf{U}), W)$ which are connected in $\mathcal{R}(\mathbf{U})$ but disconnected in $\mathsf{HGen}(\mathcal{R}(\mathbf{U}), W)$, then every path from F_1 to F_2 must contain two hyperedges whose intersection is contained in $E_1 \cap E_2$. But then every path from $\langle \Pi_{F1} \rangle_{\Pi\mathsf{Views}(\mathbf{U})}$ to $\langle \Pi_{F2} \rangle_{\Pi\mathsf{Views}(\mathbf{U})}$ must contain two hyperedges whose intersection is contained in $\langle \Pi_{E_1 \cap E_2} \rangle_{\Pi\mathsf{Views}(\mathbf{U})}$, thus implying that $(\langle \Pi_{E_1} \rangle_{\Pi\mathsf{Views}(\mathbf{U})}, \langle \Pi_{E_2} \rangle_{\Pi\mathsf{Views}(\mathbf{U})})$ is an articulation pair for $\mathsf{HGen}(\mathcal{H}(\Pi\mathsf{Views}(\mathbf{U})), N)$. Hence $\mathcal{H}(\Pi\mathsf{Views}(\mathbf{U}))$ is closed cyclic.

Conversely, assume that $\mathcal{H}(\Pi\mathsf{Views}(\mathbf{U}))$ is closed cyclic, and let N be closed block of this hypergraph. Since N is a *closed* block, it must generate a set of hyperedges of $\mathcal{H}(\Pi\mathsf{Views}(\mathbf{U}))$, and so be of the form $\bigcup\{\langle \Pi_W \rangle_{\Pi\mathsf{Views}(\mathbf{U})} \mid W \in \mathbf{W}\}$ for some $\mathbf{W} \subseteq \mathbf{U}$. Now it is easy to see that for any two hyperedges $\langle \Pi_{W_1} \rangle_{\Pi\mathsf{Views}(\mathbf{U})}$ and $\langle \Pi_{W_2} \rangle_{\Pi\mathsf{Views}(\mathbf{U})}$, we must have that $\langle \Pi_{W_1} \rangle_{\Pi\mathsf{Views}(\mathbf{U})} \cap \langle \Pi_{W_2} \rangle_{\Pi\mathsf{Views}(\mathbf{U})} = \langle \Pi_{W_1 \cap W_2} \rangle_{\Pi\mathsf{Views}(\mathbf{U})}$. In particular, if $(\langle \Pi_{W_1} \rangle_{\Pi\mathsf{Views}(\mathbf{U})}, \langle \Pi_{W_2} \rangle_{\Pi\mathsf{Views}(\mathbf{U})})$ is an articulation pair for N, then there are hyperedges $\langle \Pi_{V_1} \rangle_{\Pi\mathsf{Views}(\mathbf{U})}$ and $\langle \Pi_{V_2} \rangle_{\Pi\mathsf{Views}(\mathbf{U})}$ such that every path between them contains two hyperedges whose intersection is contained in $\langle \Pi_{W_1 \cap W_2} \rangle_{\Pi\mathsf{Views}(\mathbf{U})}$. So if we consider the closed subhypergraph $\mathsf{HGen}(\mathcal{R}(\mathbf{U}), \cup W)$ and following the bijection $\mathcal{F}_U$, we see that every path from V_1 to V_2 must contain two hyperedges whose intersection is contained in $W_1 \cap W_2$, whence (W_1, W_2) must be an articulation pair for $\cup W$. Hence $\mathcal{R}(\mathbf{\Pi}\mathsf{Views}(\mathbf{U}))$ is closed cyclic. □

The main characterization in the literature has been acyclicity rather than closed acyclicity. Using the equivalence of closed acyclicity and acyclicity, we have the following.

3.1.4 Theorem — $\mathcal{H}(\Pi\mathsf{Views}(\mathbf{U}))$ recaptures the acyclicity properties of $\mathcal{R}(\mathbf{U})$. *Let $\mathbf{R}\langle\mathbf{U}\rangle$ be any finite attribute schema. Then $\mathcal{R}(\mathbf{U})$ is acyclic iff $\mathcal{H}(\Pi\mathsf{Views}(\mathbf{U}))$ is.*

PROOF: The equivalence of acyclicity and closed acyclicity for arbitrary hypergraphs is noted in 1.5.7, and so the result follows from the previous proposition. □

Given a set-based schema $\mathbf{D}$ and a finite set of views X, by definition, $\mathcal{H}(X)$ is a hypergraph. Therefore, we may immediately utilize the abstract characterizations of acyclicity identified in Subsection 1.5 in this framework. Nonetheless, due to their importance, it is helpful to explicitly list them here.

3.1.5 Theorem — equivalent characterizations of acyclicity in the set-based context *Let $\mathbf{D}$ be a set-based schema, and let X be a finite set of views of $\mathbf{D}$. Then the following conditions are equivalent.*

(a) *$\mathcal{H}(X)$ is acyclic.*

(b) *$\mathcal{H}(X)$ is closed acyclic.*

(c) *$\mathcal{H}(X)$ is chordal and conformal.*

(d) *Graham's algorithm succeeds on* $\mathcal{H}(X)$.

(e) $\mathcal{H}(X)$ *has a join tree.*

(f) $\mathcal{H}(X)$ *has the running intersection property.*

(g) $\mathcal{H}(X)$ *has an articulated join plan.*

(h) $\mathcal{H}(X)$ *has an articulated sequential join plan.*

PROOF: Follows from 1.5.7. and 1.5.9. □

3.2 The Utility of Acyclicity in the Set-Based Case

Having established the meaning of acyclicity in the general set-based context, it is appropriate to ask of its implications. Here we show that not only is it not equivalent to the pairwise definability characterizations of Section 2, but, as a stand-alone property, it is not very interesting. We use several examples illustrate the situation.

3.2.1 Example — an anomalous acyclic schema Let $\mathbf{D}$ be a set-based schema, and let X be a finite set of views of $\mathbf{D}$ such that no two distinct members of X have commuting congruences. Then $\mathcal{H}(X) = (X, \{\{\Gamma\} \mid \Gamma \in X\})$, which is trivially acyclic. Yet we would hardly characterize such a schema as "desirable."

Thus, without some embellishment, schema acyclicity in the set-based context is not a particularly strong property. However, one might conjecture that sets of views whose congruences do not commute are not very interesting, and that as long as all of the views in a decomposition X have pairwise commuting congruences (as they do in the simple universal relational setting — see 1.3.3), acyclicity might yield something much more interesting. However, this does not appear to be the case. The next example shows that we cannot expect pairwise definability to be a consequence of acyclicity, even if we just add a few functional dependencies to an acyclic simple universal relational schema.

3.2.2 Example — an acyclic schema which is not pairwise definable Let $\mathbf{U} = \{AB, BC, CD, DE\}$, and let $\mathbf{R} = (R, \{\bowtie[\mathbf{U}], AC \to E\})$. Then it is easy to verify that there are no constraints on any of the views in $\Pi\mathsf{Views}(\mathbf{U})$; *i.e.*, $\mathsf{LDB}(\mathbf{R})[W]$ is the set of all finite subsets of $\prod_{A \in W} \mathsf{dom}(A)$. Furthermore, there are no constraints on any *pair* of views from $\Pi\mathsf{Views}(\mathbf{U})$. That is, for any $W_1, W_2 \in \mathbf{U}$, if $M_1 \in \mathsf{LDB}(\mathbf{R})[W_1]$ and $M_2 \in \mathsf{LDB}(\mathbf{R})[W_2]$ are such that ${\pi_{W_1 \cap W_2}}'(M_1) = {\pi_{W_1 \cap W_2}}'(M_2)$, then there is an $M \in \mathsf{LDB}(\mathbf{R})$ such that ${\pi_{W_1}}'(M) = M_1$ and ${\pi_{W_2}}'(M) = M_2$. It thus follows from 1.3.2 that any two views from $\Pi\mathsf{Views}(\mathbf{U})$ have commuting congruences. Yet $\Pi\mathsf{Views}(\mathbf{U})$ cannot be pairwise definable. Indeed, the quintuple $\left(\begin{bmatrix} a_1 & b_1 \end{bmatrix}, \begin{bmatrix} b_1 & c_1 \end{bmatrix}, \begin{bmatrix} c_1 & d_1 \end{bmatrix}, \begin{bmatrix} d_1 & e_1 \\ d_1 & e_2 \end{bmatrix}\right) \in \mathsf{LDB}(\mathbf{R})[AB] \times \mathsf{LDB}(\mathbf{R})[BC] \times \mathsf{LDB}(\mathbf{R})[CD] \times \mathsf{LDB}(\mathbf{R})[DE]$ is pairwise consistent but not not totally consistent. Thus acyclicity does not imply pairwise definability in the more general case, even if all pairs of views have commuting congruences.

Having pretty much dashed any hopes that acyclicity implies pairwise definability in a general context, we may ask the converse question: Does pairwise definability imply acyclicity? Again, the answer is negative, as a relatively simple example illustrates.

3.2.3 Example — a pairwise definable schema which is not acyclic Let $\mathbf{U} = \{A, B, C\}$ be a finite attribute schema with three distinct singletons, and let $\mathbf{R}$ be the simple $\mathbf{U}$-universal relational schema $(R, \{\bowtie[A, B, C]\})$. It is trivially verified that $\mathcal{R}(\mathbf{U})$ is acyclic. However, consider the set of three views $X = \{\Pi_{AB}, \Pi_{BC}, \Pi_{AC}\}$ of $\mathbf{R}$. It is easy to verify that $\mathsf{LDB}(\mathbf{R})[AB]$ consists of all finite subsets of $\mathsf{dom}(A) \times \mathsf{dom}(B)$, and similarly for the other two views. In other words, each view is constrained by the so-called *cross dependency* [27] of its two columns. It is not difficult to see that X is a subdirect decomposition of $\mathbf{R}$, and that it further is pairwise definable. Indeed, for any triple $(M_{AB}, M_{BC}, M_{AC}) \in \mathsf{LDB}(\mathbf{R})[AB] \times \mathsf{LDB}(\mathbf{R})[BC] \times \mathsf{LDB}(\mathbf{R})[AC]$, agreement on the common columns is sufficient to guarantee total consistency. Yet the hypergraph $\mathcal{H}(X)$ is cyclic. Indeed, the edges of this hypergraph are

$$H_{AB} = \{\Pi_{AB}, \Pi_A, \Pi_B, \Pi_\emptyset\}$$

$$H_{BC} = \{\Pi_{BC}, \Pi_B, \Pi_C, \Pi_\emptyset\}$$

$$H_{AC} = \{\Pi_{AC}, \Pi_A, \Pi_C, \Pi_\emptyset\}.$$

It is easily seen that Graham's algorithm fails. Hence, once we go beyond the fundamental views of a simple universal relational schema, pairwise definability no longer implies acyclicity of the underlying hypergraph.

3.2.4 Example — pairwise definability does not imply acyclicity – another perspective The astute reader will observe that, in the above example, any two of the views in $\{\Pi_{AB}, \Pi_{BC}, \Pi_{AC}\}$ form a subdirect decomposition, which is therefore in a sense redundant. However, by making things a bit more complicated, we can produce an example in which this does not occur. Specifically, let $\mathbf{U} = \{A, B, C, D, E, F\}$ be a finite attribute schema with six distinct singletons, and let $\mathbf{R}$ be the simple $\mathbf{U}$-universal relational schema $(R, \{\bowtie[A, B, C, D, E, F]\})$. We proceed as in the previous example, letting our set of views now be $X = \{\Pi_{ABD}, \Pi_{ACE}, \Pi_{BCF}\}$. We again have that X is a subdirect decomposition of $\mathbf{R}$ which is pairwise definable, and yet its hypergraph is acyclic. (Apply Graham's algorithm.) But this time, no proper subset of X forms a subdirect decomposition. Thus, even in this case, pairwise definability does not imply acyclicity.

3.3 Combining Acyclicity and Pairwise Definability

The conclusion that the examples of the preceding subsection lead us to is that acyclicity of a set-based schema hypergraph, by itself, is not a particularly strong or useful property. On the other hand, by requiring both pairwise definability and acyclicity, we obtain some very interesting generalizations of characterizations from the classical Theorem 0.1.1. In requiring both properties, we find ourselves looking at category PDA of Figures 0.1.1 and 0.1.2.

We start with a generalization of the running intersection property, which is described in 1.5.6.

3.3.1 The running meet property Let $\mathbf{D}$ be a set-based schema with X a finite set of views of $\mathbf{D}$. We say that the set of views X of the schema $\mathbf{D}$ has the *running meet property* if there is a bijection $\sigma : \{1, \ldots, \mathsf{Card}(X)\} \to X$ such that for each i, $1 < i \leq n$, there is a $j \leq i$ with $(\bigvee_{k=1}^{i-1} \sigma(k)) \mathbin{\mathbb{\wedge}} \sigma(i)) \leq \sigma(j)$. Implicit is the fact that the indicated $\mathbb{\wedge}$'s exist.

3.3.2 Proposition — Running meet generalizes running intersection. *Let* $\mathbf{U}$ *be a finite attribute schema, and let* $\mathbf{R}$ *be a simple* $\mathbf{U}$*-universal relational schema. Then* $\mathcal{R}(\mathbf{U})$ *has the running intersection property iff* $\Pi\mathsf{Views}(\mathbf{U})$ *has the running meet property.*

Proof: First assume that $\mathcal{R}(\mathbf{U})$ has the running intersection property. Let $\sigma : \{1, \ldots, \mathsf{Card}(\mathbf{U})\} \to \mathbf{U}$ be a function such that for every i, $1 < i \leq \mathsf{Card}(\mathbf{U})$, there is a $j \leq i$ with $(\bigcup_{k=1}^{i-1} \sigma(k)) \cap \sigma(i) \subseteq \sigma(j)$. From this we can deduce immediately that for every i, $1 < i \leq \mathsf{Card}(\mathbf{U})$, $(\bigvee_{k=1}^{i-1} \Pi_{\sigma(k)}) \wedge \Pi_{\sigma(i)} \leq \Pi_{\sigma(j)}$. But from the classical theorem 0.1.1 we know that the running intersection property for $\mathcal{R}(\mathbf{U})$ is equivalent to all-pairs definability (called "pairwise consistency implies total consistency" in that context) of $\Pi\mathsf{Views}(\mathbf{U})$, which is equivalent to $\Pi\mathsf{Views}(\mathbf{U})$ being completely commuting (2.1.5). But that means that we can replace "$\wedge$" with "$\mathbb{\wedge}$" to get that for every i, $1 < i \leq \mathsf{Card}(\mathbf{U})$, $(\bigvee_{k=1}^{i-1} \Pi_{\sigma(k)}) \mathbin{\mathbb{\wedge}} \Pi_{\sigma(i)} \leq \Pi_{\sigma(j)}$, which is exactly the running meet property.

Conversely, suppose that $\Pi\mathsf{Views}(\mathbf{U})$ has the running meet property. Then there is a function $\sigma : \{1, \ldots, \mathsf{Card}(\mathbf{U})\} \to \mathbf{U}$ be a function such that for every i, $1 < i \leq \mathsf{Card}(\mathbf{U})$, there is a $j \leq i$ with $(\bigcup_{k=1}^{i-1} \Pi_{\sigma(k)}) \mathbin{\mathbb{\wedge}} \Pi_{\sigma(i)} \subseteq \Pi_{\sigma(j)}$. Thus, in view of 1.3.3, we have for the same index scheme on i and j as above that $\bigvee_{k=1}^{i-1} \Pi_{\sigma(k)\cap\sigma(j)} = \bigvee_{k=1}^{i-1} \Pi_{\sigma(k)} \mathbin{\mathbb{\wedge}} \Pi_{\sigma(j)} \leq (\bigvee_{k=1}^{i-1} \Pi_{\sigma(k)}) \mathbin{\mathbb{\wedge}} \Pi_{\sigma(i)} \leq \Pi_{\sigma(j)} \leq \Pi_{\sigma(j)}$. From this we may conclude that $\Pi_{\sigma(k)\cap\sigma(i)} \leq \Pi_{\sigma(j)}$ for each appropriate i, j, and k in the above formula, whence $\sigma(k) \cap \sigma(i) \leq \sigma(j)$, and so $(\bigcup_{k=1}^{i-1} \Pi_{\sigma(i)}) \mathbin{\mathbb{\wedge}} \Pi_{\sigma(i)} \subseteq \Pi_{\sigma(j)}$, which establishes the running intersection property. □

We next pursue a generalization of join trees as defined in 1.5.5.

3.3.3 Compatibility trees — the database interpretation of join trees Let X be a set of views of a set-based schema $\mathbf{D}$. The *complete meet graph* $\mathcal{M}(X)$ for the set of views X of the schema $\mathbf{D}$ is an undirected graph whose nodes are exactly the members of X. The edge labels are taken from the set $\{\Gamma_1 \mathbin{\mathbb{\wedge}} \Gamma_2 \mid \Gamma_1, \Gamma_2 \in X \text{ and } \{\Gamma_1, \Gamma_2\} \text{ is a fully commuting pair}\}$. There is a labelled edge from Γ_1 to Γ_2 precisely in the case that $\{\Gamma_1, \Gamma_2\}$ is a fully commuting pair, in which case the label is $\Gamma_1 \wedge \Gamma_2$. Any subgraph G of $\mathcal{M}(X)$ containing all of the nodes of X is called a *meet graph* of X. For any view Γ of $\mathbf{D}$ and views $\Gamma_1, \Gamma_2 \in X$, a Γ*-path* from Γ_1 to Γ_2 is a path from Γ_1 to Γ_2 such that $\Gamma \leq \text{label}(e)$ for each edge e in the path. A meet graph G is called a *compatibility graph* if for every commuting pair $\{\Gamma_1, \Gamma_2\}$ of views from X there is a $(\Gamma_1 \mathbin{\mathbb{\wedge}} \Gamma_2)$-path from Γ_1 to Γ_2. A *compatibility tree* is a compatibility graph which is a tree.

3.3.4 Proposition — Compatibility trees generalize join trees. *Let* $\mathbf{D}$ *be a set-based schema, and let* X *be a finite set of views of* $\mathbf{D}$*. Then* X *has a compatibility tree iff it has a join tree.*

PROOF: The proof is immediate. Let K be a compatibility tree for $\Pi\mathsf{Views}(\mathbf{U})$. For each view Γ labelling a node or edge in in J, replace it with $\langle\Gamma\rangle_{\Pi\mathsf{Views}(\mathbf{U})}$. We then have a join tree for $\mathcal{R}(\mathbf{U})$. Conversely, if we start with a join tree J for $\mathcal{R}(\mathbf{U})$, each node is labelled with $\langle\Gamma\rangle_{\Pi\mathsf{Views}(\mathbf{U})}$ for some $\Gamma \in \Pi\mathsf{Views}(\mathbf{U})$, and each edge with $\langle\Gamma_1 \mathbin{\mathbb{\wedge}} \Gamma_2\rangle_{\Pi\mathsf{Views}(\mathbf{D})}$. We just drop the "$\langle\quad\rangle_{\Pi\mathsf{Views}(\mathbf{U})}$" to obtain a compatibility tree. □

Join plans in the general case are essentially the same as those of the abstract hypergraph case (1.5.8), which defines the perspective for the simple universal relational setting. We formalize this as follows.

3.3.5 Remarks on join plans Let $\mathbf{D}$ be a set-based schema, and let X be a finite set of views of $\mathbf{D}$. It is immediate that we may identify X-complete join plans for X and join plans for $\mathcal{H}(X)$. Indeed, given a join plan T for X, a node will have a label of the form $\bigvee S$, with S a set of views. We merely need replace this with S itself to obtain a join plan for $\mathcal{H}(X)$. We will therefore speak of properties of both kinds of plans interchangeably, and in particular speak of articulated join plans for sets of views.

In 2.2.2, we showed that in the simple universal relational setting, there are locally commuting join plans which are not monotone. The instance-by-instance generalization of monotonicity to the set-based setting is articulation, as shown in 3.3.7 below. (See the remarks following 2.2.3 as well.) We begin with the following lemma, which shows this condition to be true for the special case of sequential join plans, and which is critical in establishing the more general result.

3.3.6 Lemma *Let* $\mathbf{U}$ *be a finite attribute schema,* $\mathbf{R}\langle\mathbf{U}\rangle$ *a simple* $\mathbf{U}$*-universal relational schema, and let* σ *be an ordering of* $\mathbf{U}$*. Then the sequential join plan depicted in Figure 1.5.1 is monotone iff* σ *is an articulation sequence. (In that figure* $n = \mathsf{Card}(\mathbf{U})$*.)*

PROOF: First, suppose that σ is an articulation sequence. The proof that Figure 1.5.1 defines a monotone join expression is completely straightforward, and is exactly that which is used in the proof of the classical theorem 0.1.1. See, *e.g.*, [4, (10) ⇒ (12)] or [23, Lem. 13,8]. We focus on proving the converse. So, suppose that the join plan of Figure 1.5.1 is monotone. If σ is not articulated, then let i be the smallest index for which the condition $(\bigcup_{k=1}^{i-1}\sigma(k)) \cap \sigma(i) \subseteq \sigma(j)$, does not hold for any $j < i$. Let $\mathbf{V}$ be the finite attribute schema defined by the reduction of the set $\{\sigma(j) \cap \sigma(i) \mid 1 \leq j < i\}$. (That is, discard sets which are subsets of others.) Without loss of generality, assume that $\{0,1\} \subseteq \mathsf{dom}(A)$ for each $A \in \overline{\mathbf{U}}$. We define a special $\sigma(i)$-relation as follows. For each $V \in \mathbf{V}$, define $N_V = \{t : V \to \{0,1\} \mid f(A) \in \{0,1\}$ for $A \in V$ and $f(A) = 0$ otherwise$\}$. In words, N_V contains tuples whose values are unrestricted (0 or 1) on attributes in V, but are restricted to the value 0 elsewhere. Define $N = \bigcup_{V\in\mathbf{V}} N_V$. For each $U \in \mathbf{U}$, define M_U to be the U-relation $\{t \mid (\forall A \in U)(t(A) \in \{0,1\})\}$. In other words, M_U is a cartesian product of

$\{0,1\}$. Define the local $\bigcup_{k=1}^{i}\sigma(i)$ relation $R=(R_i)_{1\le i\le K}$ by

$$R_k=\begin{cases} M_{\sigma(k)} & \text{for } 1\le k\le i-1\\ N & \text{for } i=k\end{cases}$$

By construction, R is consistent. Now the relation P_{i-1} defined by $M_{\sigma(1)}\bowtie M_{\sigma(2)}\bowtie\ldots\bowtie M_{\sigma(i-1)}$ is just M_Z, where $Z=\bigcup_{m=1}^{k}\sigma(m)$, since all participating relations are cartesian products on the set $\{0,1\}$. Yet the relation P_i defined by $M_{\sigma(1)}\bowtie M_{\sigma(2)}\bowtie\ldots\bowtie M_{\sigma(i-1)}\bowtie N$ cannot satisfy $\pi_{(\bigcup_{k=1}^{i-1}\sigma(k))}{}'(P_i)=P_{i-1}$, since P_i cannot be a cartesian product, even restricted to the attributes in $\bigcup_{k=1}^{i-1}\sigma(k)$. Hence the join plan is not monotone, a contradiction. Thus, σ must be articulated, as was to be shown. □

3.3.7 Proposition — articulation generalizes monotonicity. *Let* $\mathbf{U}$ *be a finite attribute schema, and let* $\mathbf{R}\langle\mathbf{U}\rangle$ *be a simple universal relational schema. Then a join plan for* $\mathbf{R}\langle\mathbf{U}\rangle$ *is articulated iff it is monotone.*

PROOF: We can establish immediately from 1.5.9 and the classical result 0.1.1 that $\mathbf{R}$ has an articulated join plan iff it has a monotone one. However, we wish to establish the stronger result that each individual plan has one property iff it has the other. We do this with the aid of the previous lemma, which already established this result for sequential join plans.

First of all, let T be an articulated join plan for $\mathbf{R}\langle\mathbf{U}\rangle$, let n be a node of T, and let σ be an articulation ordering for $\mathbf{U}$ which has $\mathsf{Label}(n)$ as an initial set. Then, by the previous lemma, the sequential join plan defined by σ is monotone, and so in particular $\mathsf{Label}(n)$ joins completely. Since n is arbitrary, it follows that T is monotone.

Conversely, suppose that T is monotone. Let n_1 and n_2 be sibling nodes of T with common parent n. For convenience, let us write $\mathbf{W}_1$, $\mathbf{W}_2$, and $\mathbf{W}$ for the values of $\mathsf{Label}(n_1)$, $\mathsf{Label}(n_2)$, and $\mathsf{Label}(n)$), respectively, and k_1, k_2, and k for the cardinalities of $\mathbf{W}_1$, $\mathbf{W}_2$, and $\mathbf{W}$, respectively.

Since the subtrees rooted at n_1 and n_2 are themselves monotone, by the classical theorem 0.1.1 there must be articulation sequences σ_1 and σ_2 for $\mathbf{W}_1$ and $\mathbf{W}_2$, respectively. Thus, particularly, by the previous lemma, $(\sigma_2(1)\bowtie(\sigma_2(2)\bowtie(\ldots\sigma_2(k_2)))\ldots)$ is a monotone sequential join expression for $\mathbf{W}_2$. Now if $\overline{\mathbf{W}_1}\in\mathbf{W}_2$, then this is also a monotone sequential join expression for $\mathbf{W}_2\cup\{\overline{\mathbf{W}_1}\}$. If not, then we add $\overline{\mathbf{W}_1}$ on to the end to get $(\sigma_2(1)\bowtie(\sigma_2(2)\bowtie(\ldots(\sigma_2(k_2)\bowtie\overline{\mathbf{U}_1})))\ldots)$. Since our original plan T is also monotone, adding $\overline{\mathbf{W}_1}$ to the end, if necessary, preserves monotonicity. Thus, again invoking the preceding lemma, we know that the sequence obtained by adding $\overline{\mathbf{W}_1}$, if necessary, to the end of σ_2 is also an articulation sequence. But given any hyperedge H of an acyclic hypergraph $\mathbf{H}$, there is an articulation order ρ for $\mathsf{Edges}(\mathbf{H})$ with $H=\rho(1)$. (See the proof of 1.5.11.) Hence we must have an articulation sequence ρ for $\mathbf{W}_2\cup\{\overline{\mathbf{W}_1}\}$ with $\rho(1)=\overline{\mathbf{W}_1}$. But then the entire sequential join expression depicted by

$$(\sigma_1(1)\bowtie(\sigma_1(2)\bowtie(\ldots(\sigma_1(k_1)\bowtie(\rho(2)\bowtie(\rho(3)\ldots\bowtie\rho(k_2+1))))))\ldots)$$

is monotone, since $\rho(1)=\bigcup_{i=1}^{k_1}\sigma_1(i)$ and $(\sigma_1(1)\bowtie(\sigma_1(2)\bowtie(\ldots\sigma_1(k_1)))\ldots)$ is a monotone

sequential join expression, by the preceding lemma. Now the corresponding sequence

$$\sigma_1(1), \sigma_1(2), \ldots, \sigma_1(k_1), \rho(2), \rho(3), \rho(k_2 + 1)$$

may fail to be an articulation ordering because it may contain duplicates of preceding elements in the "ρ" part. However, it is clear that removing an element which is preceded in the order by a copy of itself will not destroy the monotonicity. With these duplicates removed, we may safely apply the previous lemma to conclude that the sequence is an articulation ordering. We have thus extended the articulation ordering on $\mathsf{Label}(n_1)$ to one on $\mathsf{Label}(n)$; *i.e.*, from child to parent. Repeating this process, we can extend the articulation ordering on $\mathsf{Label}(n_1)$ to one on the set of hyperedges labelling the root, which must be $\mathbf{U}$. Since n_1 is an arbitrary non-root node, it follows that T is articulated, as required. □

We are now in a position to state and prove our main theorem regarding properties involving both acyclicity and pairwise definability.

3.3.8 Theorem — characterization of combined pairwise definability and acyclicity *Let $\mathbf{D}$ be a set-based schema, and let X be a finite set of views of $\mathbf{D}$. Then the following conditions are equivalent.*

(a) *X is pairwise definable and $\mathcal{H}(X)$ is acyclic.*

(b) *X is pairwise definable and has a compatibility tree.*

(c) *X has the running meet property.*

(d) *X has an articulated fully commuting X-complete join plan.*

(e) *X has an articulated fully commuting X-complete sequential join plan.*

PROOF: The equivalence (a) ⇔ (b) follows immediately from the preceding lemma and 3.1.6. The equivalence of (a), (d), and (e) follows from 1.5.9 and 2.2.3. The equivalence of (b) and (c) are established in propositions below. □

It is interesting to note that the characterization involving the running meet property is in a sense distinguished. While (a), (b), (d), and (e) all have the structure of combining a characterization of pairwise definability with a characterization of acyclicity, the running meet property (c) stands on its own as a characterization of their combination.

We now provide the promised proofs for the above theorem.

3.3.9 Proposition — ((a) ⇒ (c) of 3.3.8) *Let $\mathbf{D}$ be a set-based schema, and let X be a finite set of views of $\mathbf{D}$. Then, if X is pairwise definable and $\mathcal{H}(X)$ is acyclic, it also has the running meet property.*

PROOF: Assume that X is both pairwise definable and acyclic. From 3.1.6, we know that $\mathcal{H}(X)$ has the running intersection property. Thus, there is a bijection $\sigma : \{1, \ldots, \mathsf{Card}(X)\} \to X$ such that for every i, $1 < i \leq \mathsf{Card}(X)$, there is a $j \leq i$ with $(\bigcup_{k=1}^{i-1} \langle \sigma(k) \rangle_X) \cap \langle \sigma(i) \rangle_X \subseteq \langle \sigma(j) \rangle_X$. From pairwise definability, we also know that X is completely commuting (2.1.5);

therefore $(\bigcup_{k=1}^{i-1}\langle\sigma(k)\rangle_X) \cap \langle\sigma(i)\rangle_X = \bigcup_{k=1}^{i-1}(\langle\sigma(k)\rangle_X \cap \langle\sigma(i)\rangle_X) = \bigcup_{k=1}^{i-1}\langle\sigma(k)\rangle_X \Join \langle\sigma(i)\rangle_X$. But this means that for each k, $1 \leq k \leq i-1$, we have that $\sigma(k) \Join \sigma(i) \leq \sigma(j)$, and so $\bigvee_{k=1}^{i-1}(\sigma(k) \Join \sigma(i)) \leq \sigma(j)$, and invoking complete commutativity once again, we have $(\bigvee_{k=1}^{i-1}\sigma(k)) \Join \sigma(i) \leq \sigma(j)$, which is exactly the condition required for the running meet property. □

3.3.10 Lemma *Let* **D** *be a set-based schema, and let* X *be a finite set of views of* **D**. *Then, if* X *has the running meet property, it also has an* X*-complete fully commuting sequential join plan.*

PROOF: The proof is very similar in to that of the classical case for simple universal relational schemata [4]. Specifically, assume that X has the running meet property, and let $\sigma : \{1, \ldots, \mathsf{Card}(X)\} \to X$ be the bijection such that for each i, $1 < i \leq n$, there is a $j \leq i$ with $(\bigvee_{k=1}^{i-1}\sigma(k)) \Join \sigma(i)) \subseteq \sigma(j)$. It is immediate that the tree depicted in Figure 1 represents an X-complete fully commuting sequential generalized join plan. □

3.3.11 Proposition — ((c) ⇒ (a) of 3.3.8) *Let* **D** *be a set-based schema, and let* X *be a finite set of views of* **D**. *Then, if* X *has the running meet property, it is also pairwise definable and* $\mathcal{H}(X)$ *is acyclic.*

PROOF: Assume that X has the running meet property. From the preceding lemma, we know that X has an X-complete fully commuting sequential join plan, and hence by 2.2.3, X is pairwise definable. Now, reversing the argument of the proof of 3.3.9 above, we have that the running meet property yields the running intersection property, and so $\mathcal{H}(X)$ is acyclic by 1.5.7. □

4. Characterizations of Simplicity Involving Order Properties

We now turn to the final category of decompositions identified in Figure 0.1.1 — SFR, the existence of a sequentially definable full reducer. As illustrated in Figure 0.1.1, the generalization of this property seems to stand alone, properly between those of equivalence to pairwise definability identified in Section 2 and those of equivalence to pairwise definability plus acyclicity identified in Section 3.

4.1 Principles of Order-Based Schemata

Because of the very nature of this characterization, it is necessary to work with an environment with more structure than the the simple set-based one. Specifically, to "reduce" a tuple of states to a consistent one, we need a concept of size. Therefore, we begin by identifying the properties of the framework in which this characterization will be formulated.

4.1.1 Full reducers in the relational case To motivate the need for an order structure, let us begin by recalling the notion of a full reducer in the context of a simple universal relational schema. Let $\mathbf{U}$ be a finite attribute schema, with $\mathbf{R}\langle\mathbf{U}\rangle$ a simple $\mathbf{U}$-universal relational schema. Suppose further that we are given a state $\vec{M} = (M_W)_{W \in \mathbf{U}} \in \{\Pi_W \mid W \in \mathbf{U}\}$, but it is not necessarily the case that $\vec{M} \in \Delta\langle\mathbf{U}\rangle'(\mathsf{LDB}(\mathbf{R}\langle\mathbf{U}\rangle))$. In other words, $\vec{M}$ may not be a decomposed database. It is easy to see that there is a largest (under relation-by-relation inclusion) $\vec{N} = (N_W)_{W \in \mathbf{U}} \in \prod_{W \in \mathbf{U}} \mathsf{LDB}(\mathbf{R})[W]$ such that $\vec{N} \subseteq \vec{M}$ and $\vec{N} \in \Delta\langle\mathbf{U}\rangle'(\mathsf{LDB}(\mathbf{R}))$. Indeed, $\vec{N}$ if found by simply joining the M_i's and discarding those tuples which do not join; it is called the *full reduction* of $\vec{M}$. A *sequential full reducer* (often called just a *full reducer* in the literature) is a program which computes this $\vec{N}$ in a nice way. More precisely, the *semijoin* operation $W_1 \ltimes W_2$ applied to a pair $(M_{W_1}, M_{W_2}) \in \mathsf{LDB}(\mathbf{R})[W_1] \times \mathsf{LDB}(\mathbf{R})[W_2]$ yields ${\pi_{W_1}}'(M_{W_1} \bowtie M_{W_2})$; this result is often denoted by $M_{W_1} \ltimes M_{W_2}$. A *(sequential) full reducer* is a sequence of replacement operations of the form $M_{W_1} \leftarrow M_{W_1} \ltimes M_{W_2}$. The classical result (condition (S8) of Theorem 0.1) states that we may compute the largest compatible state ($\vec{N}$ in the above discussion) with a sequential full reducer iff $\mathbf{R}\langle\mathbf{U}\rangle$ is acyclic.

4.1.2 Ordered schemata and views To generalize the notion of reduction to the set-based context, we must incorporate a notion of relative database size into our model. This is accomplished by augmenting set-based schemata with an order structure. Specifically, an *ordered database schema* is a set-based schema $\mathbf{D}$ equipped with a partial order $\leq$ on $\mathsf{LDB}(\mathbf{D})$. Any relational schema may be regarded as an ordered schema under relation-by-relation inclusion. On a product schema $\prod_{i \in I} \mathbf{V}_i$ indexed by the set I we use the product ordering; *i.e.*, $(M_i)_{i \in I} \leq (N_i)_{i \in I}$ iff $M_i \leq N_i$ for all $i \in I$. An *order view* $\Gamma = (\mathbf{V}_\Gamma, \mu_\Gamma)$ of $\mathbf{D}$ is a view with the property that $\mathbf{V}_\Gamma$ is an order schema and ${\mu_\Gamma}'$ is an order-preserving function; *i.e.*, for any $M, N \in \mathsf{LDB}(\mathbf{D})$, $M \leq N$ implies that ${\mu_\Gamma}'(M) \leq {\mu_\Gamma}'(N)$.

4.2 Semireducers in the General Setting

We now turn to defining the generalization of a full reducer to the context of a general set-based schema with order structure.

4.2.1 Full and semi- reduction In order to formulate a generalization of full reduction, we need a further property on decompositions. Let X be a finite set of order views of the ordered database schema $\mathbf{D}$. We say that X has the *maximal matching property* if it satisfies the following three properties.

(mmp-i) Each $\Gamma \in X$ is an order view of $\mathbf{D}$.

(mmp-ii) Each pair of views $\{\Gamma_1, \Gamma_2\} \subseteq X$ is commuting.

(mmp-iii) For any nonempty subset $Y \subseteq X$ and any $\vec{M} = (M_\Gamma)_{\Gamma \in Y} \in \prod_{\Gamma \in Y} \mathsf{LDB}(\mathbf{V}_\Gamma)$, there is a greatest $\vec{N} = (N_\Gamma)_{\Gamma \in Y} \in \Delta\langle Y\rangle'(\mathsf{LDB}(\mathbf{D}))$ with the property that $\vec{N} \leq \vec{M}$.

$\vec{N}$ is called the full Y-reduction of $\vec{M}$, and is denoted $\mathsf{FullRed}(\vec{M}, Y)$. The component of $\mathsf{FullRed}(\vec{M}, Y)$ which is indexed by the view $\Gamma \in Y$ is denoted $\mathsf{FullRed}(\vec{M}, Y, \Gamma)$.

Now assume that X has the maximal matching property, let Γ_1 and Γ_2 be elements of X, and let $\vec{M} = (M_\Gamma)_{\Gamma \in X} \in \prod_{\Gamma \in X} \mathsf{LDB}(\mathbf{V}_\Gamma)$. The (Γ_1, Γ_2)-*semireduction of* $\vec{M}$, denoted $\mathsf{SemiRed}(\vec{M}, (\Gamma_1, \Gamma_2))$, is the tuple $\vec{N} = (N_\Gamma)_{\Gamma \in X}$ defined by

$$N_\Gamma = \begin{cases} \mathsf{FullRed}(\vec{M}, \{\Gamma_1, \Gamma_2\}, \Gamma_1) & \text{if } \Gamma = \Gamma_1; \\ M_\Gamma & \text{otherwise.} \end{cases}$$

If $\Gamma \in X$, we denote by $\mathsf{SemiRed}(\vec{M}, (\Gamma_1, \Gamma_2), \Gamma)$ the component of $\mathsf{SemiRed}(\vec{M}, (\Gamma_1, \Gamma_2))$ indexed by Γ.

Finally, given a finite sequence $S = (\Gamma_{11}, \Gamma_{12}), \ldots, (\Gamma_{m1}, \Gamma_{m2}) \in (X \times X)^*$, define, for $0 \leq j \leq m$ and $\vec{M} \in \prod_{W \in \mathbf{U}} \mathsf{LDB}(\mathbf{R})[W]$,

$$\mathsf{Seqred}\langle S \rangle_j(\vec{M}) = \begin{cases} \vec{M} & \text{if } j = 0; \\ \mathsf{SemiRed}(\mathsf{Seqred}\langle S \rangle_{j-1}, (\Gamma_{j1}, \Gamma_{j2})) & \text{if } j > 0. \end{cases}$$

Define $\mathsf{Seqred}\langle S \rangle(M) = \mathsf{Seqred}\langle S \rangle_m(M)$, with m the length of the sequence S, and for $\Gamma \in X$ define $\mathsf{Seqred}\langle S, \Gamma \rangle(M)$ to be the component indexed by Γ. The above formula just formalizes the application of the semireductions defined by S to $\vec{M}$ in a sequential fashion. The sequence S is a *semireducer-based full reducer* for X if $\mathsf{Seqred}\langle S \rangle(M) = \mathsf{FullRed}(M, X)$ for every $\vec{M} \in \prod_{\Gamma \in X} \mathsf{LDB}(\mathbf{V}_\Gamma)$. This definition requires that both $\mathsf{FullRed}(\vec{M}, X)$ always exist, and that it be computable by a sequence of pairwise reductions. We say that X *admits a semireducer-based full reducer* if such an S exists.

It is clear that these concepts generalize the corresponding ones in the simple universal relational case. Note in particular that any projection of a simple universal relational schema is necessarily an order view, and that by 1.3.3, any pair of projection is commuting. We therefore have the following result.

4.2.2 Proposition *Let* $\mathbf{U}$ *be a finite attribute schema, and let* $\mathbf{R}\langle \mathbf{U} \rangle$ *be a simple* $\mathbf{U}$*-universal relational schema. Then* $\mathbf{R}\langle \mathbf{U} \rangle$ *admits a full reducer (in the traditional sense) iff* $\Pi\mathsf{Views}(\mathbf{U})$ *admits a semireducer-based full reducer.* □

We are now in a position to state and prove our main characterization. As we have done in previous sections, for pedagogical reasons, we first state the main result, and then prove it via a series of lemmata and propositions.

4.2.3 Theorem *Let* $\mathbf{D}$ *be an ordered database schema, and let* X *be a finite set of order views of* $\mathbf{D}$ *with the maximal matching property. Consider the following properties.*

(a) X *is pairwise definable and* $\mathcal{H}(X)$ *is acyclic.*

(b) X *admits a semireducer-based full reducer.*

(c) X *is pairwise definable.*

Then (a) ⇒ (b) ⇒ (c)*, but none of the reverse implications hold.*

PROOF: The implication (a) ⇒ (b) is proved in 4.2.7 below, while (b) ⇒ (c) is proved in 4.2.9. The failures of (b) ⇒ (a) and (c) ⇒ (b) are established with examples in 4.2.8 and 4.2.10, respectively. □

4.2.4 Lemma *Let* $\mathbf{D}$ *be an ordered database schema, and let* X *be a finite set of views of* $\mathbf{D}$ *with the maximal matching property. Further, let* $\vec{M} \in \prod_{\Gamma \in X} \mathsf{LDB}(\mathbf{V}_\Gamma)$*, and let* $(\Gamma_1, \Gamma_2) \in X \times X$ *with* $\{\Gamma_1, \Gamma_2\}$ *a commuting pair. Then, for any* $\vec{M} = (M_\Gamma)_{\Gamma \in X} \in \prod_{\Gamma \in X} \mathsf{LDB}(\mathbf{V}_\Gamma)$*, we have that* $\lambda(\Gamma_1, \Gamma_2)'(\mathsf{SemiRed}(\vec{M}, (\Gamma_1, \Gamma_2), \Gamma_1) \leq \lambda(\Gamma_2, \Gamma_1)'(M_{\Gamma_2})$.

PROOF: It is clear that we must have $\lambda(\Gamma_1, \Gamma_2)'(\mathsf{SemiRed}(\vec{M}, (\Gamma_1, \Gamma_2), \Gamma_1) = \lambda(\Gamma_2, \Gamma_1)'(\mathsf{SemiRed}((\mathsf{SemiRed}(\vec{M}, (\Gamma_1, \Gamma_2), \Gamma_1), (\Gamma_2, \Gamma_1), \Gamma_2)$, as any decomposed database must have agreement on the meets of its components. But $\lambda(\Gamma_2, \Gamma_1)'(\mathsf{SemiRed}(\mathsf{SemiRed}(\vec{M}, (\Gamma_1, \Gamma_2), \Gamma_1), (\Gamma_2, \Gamma_1), \Gamma_2)) \leq \lambda(\Gamma_2, \Gamma_1)'(M_{\Gamma_2})$, whence the result. □

4.2.5 Lemma *Let* $\mathbf{D}$ *be an ordered database schema, and let* X *be a finite set of views of* $\mathbf{D}$ *with the maximal matching property. Further, let* $\vec{M} \in \prod_{\Gamma \in X} \mathsf{LDB}(\mathbf{V}_\Gamma)$*, and let* S *be any finite sequence of elements from* $X \times X$*. Then* $\mathsf{FullRed}(\vec{M}, X) \leq \mathsf{Seqred}\langle S \rangle(\vec{M})$*. Thus, any sequence* S *for which* $\mathsf{Seqred}\langle S \rangle(\vec{M}) \in \Delta\langle X \rangle'(\mathsf{LDB}(\mathbf{D})$ *for all* $\vec{M} \in \prod_{\Gamma \in X} \mathsf{LDB}(\mathbf{V}_\Gamma)$ *must in fact be a semireducer-based full reducer.*

PROOF: It is immediate that $\mathsf{FullRed}(\vec{M}, X) \leq \mathsf{SemiRed}(\vec{M}, (\Gamma_1, \Gamma_2))$ for any single semireduction. But then we must have that $\mathsf{FullRed}(\vec{M}, X) = \mathsf{FullRed}(\mathsf{SemiRed}(\vec{M}, (\Gamma_1, \Gamma_2)), X)$, whence the result follows by a simple induction. □

The running meet property on a set of views X states (among other things) that we can line the views up (represented by an ordering α) such that each view (indexed by) i has a distinguished predecessor $\beta(i)$ with the property that for any other predecessor $\alpha(k)$ of $\alpha(i)$, $\alpha(i) \mathbb{\wedge} \alpha(k) \leq \alpha(i) \mathbb{\wedge} \alpha(k)$. The next rather technical lemma states that to verify pairwise compatibility of a local X-database $\vec{M}$, it suffices to check compatibility of each view $\alpha(i)$ with its distinguished predecessor $\beta(i)$. Note that this result holds in the general set-based framework, and does not require any order properties.

4.2.6 Lemma *Let* $\mathbf{D}$ *be a set-based schema, and let* X *be a finite set of views of* $\mathbf{D}$ *with the running meet property. As guaranteed by the running meet property, let* $\alpha : \{1, \ldots, \mathsf{Card}(X)\} \to X$ *be a bijection such that for each* i*,* $1 < i \leq \mathsf{Card}(X)$*, there is a* $j \leq i$ *with* $(\bigvee_{k=1}^{i-1} \alpha(k)) \mathbb{\wedge} \alpha(i)) \leq \alpha(j)$*. For each* i*,* $2 \leq i \leq \mathsf{Card}(X)$*, let* $\beta(i) \in X$ *denote a specific view in* $\{\alpha(k) \mid 1 \leq k \leq i\}$ *with the property that* $(\bigvee_{k=1}^{i-1} \alpha(k)) \mathbb{\wedge} \alpha(i) \leq \beta(i)$*. Then, if* $\vec{M} = (M_\Gamma)_{\Gamma \in X} M \in \prod_{\Gamma \in X} \mathsf{LDB}(\mathbf{V}_\Gamma)$ *has the property that* $\lambda(\beta(i), \alpha(i) \mathbb{\wedge} \beta(i))'(M_{\beta(i)}) = \lambda(\alpha(i), \alpha(i) \mathbb{\wedge} \beta(i))'(M_{\alpha(i)})$ *for all* i*,* $2 \leq i \leq \mathsf{Card}(X)$*,* $\vec{M}$ *is pairwise compatible.*

PROOF: With α and β as defined above, we assume that $\vec{M} = (M_W)_{W \in \mathbf{U}} \in \prod_{\Gamma \in X} \mathsf{LDB}(\mathbf{V}_\Gamma)$ has the property that $\lambda(\beta(i), \alpha(i) \mathbb{\wedge} \beta(i))'(M_{\beta(i)}) = \lambda(\alpha(i), \alpha(i) \mathbb{\wedge} \beta(i))'(M_{\alpha(i)})$ for all i, $2 \leq i \leq$

$\mathsf{Card}(X)$. For each integer n in the range $1 \leq n < card(X)$, let $\varphi(n)$ be the statement that for each $1 \leq i, j \leq n$, we have that $\lambda(\alpha(i), \alpha(i)\mathbb{\wedge}\alpha(j))(M_{\alpha(i)}) = \lambda(\alpha(j), \alpha(i)\mathbb{\wedge}\alpha(j))(M_{\alpha(j)})$. We proceed by induction on n to prove that $\varphi(n)$ is true for all n, $2 \leq n \leq \mathsf{Card}(X)$. As the basis, note that we must have $\beta(2) = \alpha(1)$, so that $\lambda(\alpha(1), \alpha(1)\mathbb{\wedge}\alpha(2))(M_{\alpha(1)}) = \lambda(\alpha(2), \alpha(1)\mathbb{\wedge}\alpha(2))(M_{\alpha(2)})$. Now let $1 \leq n < \mathsf{Card}(X)$, and assume that $\varphi(i)$ is true for all $i < n$. From $(\bigvee_{k=1}^{n-1} \alpha(k))\mathbb{\wedge}\alpha(n) \leq \beta(n)$ we know that $(\bigvee_{k=1}^{n-1}(\alpha(k)\mathbb{\wedge}\alpha(n))) \leq \beta(n)$, since $\mathbb{\wedge}$ is just the meet in the weak partial lattice $[\mathbf{CView}(\mathbf{D})]$. But then for each $k \leq n$, $\alpha(k)\mathbb{\wedge}\alpha(n) \leq \beta(n)$, and this implies that $\alpha(k)\mathbb{\wedge}\alpha(n) \leq \alpha(k)\mathbb{\wedge}\beta(n)$, so $\lambda(\alpha(n), \alpha(n)\mathbb{\wedge}\alpha(k))'(M_{\alpha(n)}) = \lambda(\alpha(n)\mathbb{\wedge}\beta(n), \alpha(n)\mathbb{\wedge}\beta(n)\mathbb{\wedge}\alpha(k))' \circ \lambda(\alpha(n), \alpha(n)\mathbb{\wedge}\beta(n))'(M_{\alpha(n)}) = \lambda(\alpha(n)\mathbb{\wedge}\beta(n), \alpha(n)\mathbb{\wedge}\beta(n)\mathbb{\wedge}\alpha(k))' \circ \lambda(\beta(n), \alpha(n)\mathbb{\wedge}\beta(n))'(M_{\beta(n)}) = \lambda(\beta(n), \alpha(n)\mathbb{\wedge}\beta(n)\mathbb{\wedge}\alpha(k))'(M_{\beta(n)}) = \lambda(\alpha(k), \alpha(n)\mathbb{\wedge}\beta(n)\mathbb{\wedge}\alpha(k))'(M_{\alpha(k)}) = \lambda(\alpha(k), \alpha(n)\mathbb{\wedge}\alpha(k))'(M_{\alpha(k)})$. Thus, $\alpha(n)$ is pairwise compatible with all views in $\{\alpha(k) \mid 1 \leq k \leq n-1\}$, and so $\varphi(n)$ is true. The inductive proof is therefore complete. □

4.2.7 Proposition — ((a) ⇒ (b) in 4.2.2) *Let* $\mathbf{D}$ *be a set-based schema, and let* X *be a finite set of views of* $\mathbf{D}$*. Assume further that* X *has the maximal matching property. Then, if* X *has the running meet property, it also admits a semireducer-based full reducer.*

PROOF: As guaranteed by the running meet property, let $\alpha : \{1, \ldots, \mathsf{Card}(X)\} \to X$ be a bijection such that for each i, $1 < i \leq n$, there is a $j \leq i$ with $(\bigvee_{k=1}^{i-1} \alpha(k))\mathbb{\wedge}\alpha(i)) \leq \alpha(j)$. For each i, $2 \leq i \leq \mathsf{Card}(X)$, let $\beta(i) \in X$ denote a specific view in $\{\alpha(k) \mid 1 \leq k \leq i\}$ with the property that $(\bigvee_{k=1}^{i-1} \alpha(k))\mathbb{\wedge}\alpha(i)) \leq \beta(i)$. Define the sequence S by the following program.

```
1.  S ← ();
2.  for  i ← Card(X) downto 2 do
3.       for j ← 1 to i − 1 do
4.            S ← S · (α(j), α(i))
5.       end do;
6.  end do;
7.  for  i ← 2 to Card(X) do
8.       S ← S · (α(i), β(i))
9.  end do;
```

We claim that S is a semireducer-based full reducer for X. Let $S_{\text{1-6}}$ denote the subsequence of S computed by lines 1-6 of the above program. Since X has the running meet property, it is also completely commuting (2.1.5). Therefore, we may apply 4.2.4 to conclude that for any i and j with $\alpha(j) \leq \alpha(i)$ and for any $\vec{M} = (M_\Gamma)_{\Gamma \in Y} \in \prod_{\Gamma \in Y} \mathsf{LDB}(\mathbf{V}_\Gamma)$, we have that $\lambda(\alpha(j), \alpha(i)\mathbb{\wedge}\alpha(j))'(\mathsf{Seqred}\langle S_{16}, \alpha(j)\rangle(\vec{M})) \leq \lambda(\alpha(i), \alpha(i)\mathbb{\wedge}\alpha(j))'(\mathsf{Seqred}\langle S_{16}, \alpha(i)\rangle(\vec{M}))$. Now consider the action of the semireductions represented by the pairs added to S in lines 7-9 of the program. For each view $\alpha(i)$, we compute the semireduction with that single view $\beta(i)$ to the left (in the ordering defined by α) with the property that $(\bigvee_{k=1}^{i-1} \alpha(k))\mathbb{\wedge}\alpha(i)) \leq \beta(i)$.

But in view of the preceding lemma, the resulting tuple will be pairwise compatible; *i.e.*, $\mathsf{Seqred}\langle S\rangle(\vec{M}) \in \Delta\langle X\rangle'(\mathsf{LDB}(\mathbf{D}))$. Finally, 4.2.5 guarantees that S must in fact be a full reducer, as required. □

As we stated in 4.2.3, the implication (**b**) ⇒ (**a**) does not hold. We now provide a specific counterexample.

4.2.8 Example — A pairwise definable schema with no semireducer-based full reducer Let $\mathbf{U}$ be the finite attribute schema $\{AB, AC, BC\}$, and let $\mathbf{R}$ be the relational schema $(\{R_{AB}, R_{BC}, R_{AC}\}, \{R_{AB}[A] = R_{AC}[A], R_{AB}[B] = R_{BC}[B], R_{AC}[C] = R_{BC}[C]\})$. Here R_{AB} is a binary relation on the attributes AB, R_{AC} is a binary relation on the attributes AC, and R_{BC} is a binary relation on the attributes BC. An instance is thus a triple of relations. The order relation on $\mathbf{R}$ is relation-by-relation inclusion.

The schema is constrained by three dependencies. $R_{AB}[A] = R_{AC}[A]$ means that in any legal database, the A-projection of the relation associated with R_{AB} must agree with the A-projection of the relation associated with R_{AC}. The dependencies $R_{AB}[B] = R_{BC}[B]$ and $R_{AC}[C] = R_{BC}[C]$ have their semantics defined similarly. Each of these dependencies is a combination of two *inclusion dependencies* [6]. Let X consist of the three views $\Upsilon_{R_{AB}}$, $\Upsilon_{R_{AC}}$, and $\Upsilon_{R_{BC}}$. The view $\Upsilon_{R_{AB}}$ has as its underlying schema the simple universal relational schema $\mathbf{R}_{AB} = (R_{AB}, \emptyset)$ over the attribute schema $\{AB\}$. The associated view morphism $\mu_{\Upsilon_{R_{AB}}}$ preserves the instance of R_{AB} exactly but discards the instances of R_{AC} and R_{BC}. The view schemata and morphisms for $\Upsilon_{R_{AC}}$ and $\Upsilon_{R_{BC}}$ are defined analogously by preserving the relation corresponding to their subscripts, and discarding the other two. It is easily verified that X consists of order views and has the maximal matching property. For any triple $\vec{M} = (M_{AB}, M_{AC}, M_{BC}) \in \mathsf{LDB}(\mathbf{R}_{AB}) \times \mathsf{LDB}(\mathbf{R}_{AC}) \times \mathsf{LDB}(\mathbf{R}_{BC})$, $\mathsf{FullRed}(\vec{M}, X)$ is obtained by joining the three relations together and then recomputing the three projections. X is furthermore pairwise definable — we just need check agreement on the common columns.

Let us show that X does not admit a semireducer-based full reducer. To this end, pick any positive number $n > 3$, and let $\{a_1, \ldots a_{n+1}\} \subseteq \mathsf{dom}(A)$, $\{b_1, \ldots b_{n+1}\} \subseteq \mathsf{dom}(B)$, $\{c_1, \ldots c_{n+1}\} \subseteq \mathsf{dom}(C)$, be sets of $n+1$ distinct elements from these domains. We define the following instances.

$$M_{AB} = \begin{pmatrix} a_1 & b_1 \\ a_2 & b_2 \\ \vdots & \vdots \\ a_n & b_n \end{pmatrix} \qquad M_{AC} = \begin{pmatrix} a_1 & c_{n+1} \\ a_2 & c_1 \\ a_3 & c_2 \\ \vdots & \vdots \\ a_{n+1} & c_n \end{pmatrix} \qquad M_{BC} = \begin{pmatrix} b_1 & c_1 \\ b_2 & c_2 \\ \vdots & \vdots \\ b_{n+1} & c_{n+1} \end{pmatrix}$$

It is easy to see that the full reduction of (M_{AB}, M_{AC}, M_{BC}) is the triple $(\emptyset, \emptyset, \emptyset)$ of empty relations. On the other hand, any two of these relations can be made pairwise consistent by discarding exactly one tuple from each, and this property is preserved after discarding such a tuple. It follows that any semireducer-based full reducer must perform at least $3 \cdot (n+1)$ semireductions to finally arrive at the full reduction. Since n is arbitrary, it follows that no finite sequence $S \in (X \times X)^*$ can define a semireducer-based full reducer. Hence $\mathbf{R}$ is does not have a semireducer-based full reducer.

4.2.9 Proposition — ((b) ⇒ (c) in 4.2.2) *Let* $\mathbf{D}$ *be a set-based schema, and let* X *be a finite set of views of* $\mathbf{D}$*. Assume further that* $\mathbf{D}$ *has the maximal matching property. Then, if* X *admits a semireducer-based full reducer, it is pairwise definable.*

PROOF: Let $\vec{M} = (M_\Gamma)_{\Gamma \in X} \in \prod_{\Gamma \in X} \mathsf{LDB}(\mathbf{V}_\Gamma)$ be pairwise consistent. Then, for any pair $(\Gamma_1, \Gamma_2) \in X \times X$, we must have that $\mathsf{SemiRed}(\vec{M}, (\Gamma_1, \Gamma_2), \Gamma_1) = \vec{M}$. Thus, $\vec{M}$ is a fixpoint of any program of the form $\mathsf{Seqred}\langle S \rangle$ for any finite sequence $S \in (X \times X)^*$. In particular, it is a fixpoint of any semireducer-based full reducer. Thus, if such a semireducer-based full reducer exists, $\vec{M}$ is globally consistent. In other words, if X has a semireducer-based full reducer, every pairwise consistent local X-database is globally consistent, and so X is pairwise definable. □

Finally, we provide a counterexample to (c) ⇒ (b) of 4.2.3.

4.2.10 Example — A schema admitting a semireducer-based full reducer whose hypergraph is not acyclic Consider again the schema $\mathbf{R}$ and set of views X defined in 3.3.3, which was shown there to have a cyclic hypergraph. We regard each relational schema as an order schema under relation-by-relation inclusion. X admits the maximal model property — we obtain the full reduction of $\vec{M} = (M_{AB}, M_{BC}, M_{AC}) \in \mathsf{LDB}(\mathbf{R})[AB] \times \mathsf{LDB}(\mathbf{R})[BC] \times \mathsf{LDB}(\mathbf{R})[AC]$, by computing the join and then recomputing the three projections. The semireduction $\mathsf{SemiRed}(\vec{M}, (\Pi_{AB}, \Pi_{AC}), \Pi_{AB})$ retains just those tuples of M_{AB} which join with some tuple of M_{AC}. Note that unless there are no matches at all on the corresponding columns identified by attribute A, $\mathsf{SemiRed}(\vec{M}, (\Pi_{AB}, \Pi_{AC}), \Pi_{AB})$ and M_{AB} will have exactly the same A-projections. Similar observations hold for the other five semireduction possibilities. In short, after computing (in any order) each of the six semireductions once, we are left with a triple $\vec{N} = (N_{AB}, N_{BC}, N_{AC})$ which is either pairwise compatible or else contains at least one empty relation. In the former case, we have already computed the full reduction. In the latter case, one more pass of each of the six possible semireductions will yield the full reduction, which is the triple of empty relations. Thus, a semireducer based full reducer S is obtained by letting $S_o \subseteq (X \times X)^*$ be any sequence which contains each element of $X \times X$ exactly once, and then setting $S = S_o \cdot S_o$.

5. Summary, Conclusions, and Further Directions

5.1 Summary and Conclusions

We have presented an extensive classification of properties of decompositions of set-based database schemata, generalizing and augmenting the classical relational theory. Because of the large number of characterizations (28 in all), it is appropriate to step back and present a summary of the overall picture. In essence, we identify the members of the equivalence classes identified in Figures 0.1.1 and 0.1.2.

5.1.1 The classes of schema simplicity Let $\mathbf{D}$ be a set-based database schema, and let X be a finite set of views of $\mathbf{D}$. Define the following classes of properties.

The class $\mathsf{PD}(X)$ consists of the following statements.

- X is pairwise definable.
- X is strongly pairwise definable.
- X is all-pairs definable.
- X is all-pairs strongly definable.
- X satisfies the generalized Chinese remainder condition.
- X is pairwise commuting and $\mathsf{Span}(X, [\mathbf{View}(\mathbf{D})])$ is a distributive sublattice of $[\mathbf{View}(\mathbf{D})]$.
- $\mathsf{Span}(X, [\mathbf{CView}(\mathbf{D})])$ is a distributive sublattice of $[\mathbf{CView}(\mathbf{D})]$.
- X is completely commuting.
- X has an X-complete locally commuting join plan.
- X has an X-complete locally commuting sequential join plan.
- X is decomposition equivalent to a set of embedded CBD's.
- X is decomposition equivalent to a set of conflict-free embedded CBD's.
- X is decomposition equivalent to a set of X-complete CBD's.
- X is decomposition equivalent to a set of conflict-free X-complete CBD's.

The class $\mathsf{ACY}(X)$ consists of the following statements.

- $\mathcal{H}(X)$ is acyclic.
- $\mathcal{H}(X)$ is closed acyclic.
- $\mathcal{H}(X)$ is chordal and conformal.
- Graham's algorithm succeeds on $\mathcal{H}(X)$.
- $\mathcal{H}(X)$ has a join tree.
- $\mathcal{H}(X)$ has the running intersection property.
- $\mathcal{H}(X)$ has an articulated join plan.
- $\mathcal{H}(X)$ has an articulated sequential join plan.

The class $\mathsf{PDA}(X)$ consists of the following statements.

- X is pairwise definable and $\mathcal{H}(X)$ is acyclic.
- X is pairwise definable and has a compatibility tree.
- X has the running meet property.
- X has an articulated fully commuting X-complete join plan.

- X has an articulated fully commuting X-complete sequential join plan.

The class $\mathsf{SFR}(X)$ consists of the following statement.

- X admits a semireducer-based full reducer.

Here, then, is a summary of our main results.

5.1.2 Theorem *Let* $\mathbf{D}$ *be a set-based database schema, and let* X *be a finite set of views of* $\mathbf{D}$. *We then have the following.*

(a) *If* X *satisfies any property in* $\mathsf{PD}(X)$, *then it satisfies every property in* $\mathsf{PD}(X)$.

(b) *If* X *satisfies any property in* $\mathsf{ACY}(X)$, *then it satisfies every property in* $\mathsf{ACY}(X)$.

(c) *If* X *satisfies any property in* $\mathsf{PDA}(X)$, *then it satisfies every property in* $\mathsf{PDA}(X)$, $\mathsf{PD}(X)$, *and* $\mathsf{ACY}(X)$.

(d) *If* $\mathbf{D}$ *is an ordered database schema,* X *is a finite set of order views with the maximal matching property, and* X *satisfies any property in* $\mathsf{PDA}(X)$, *then it also satisfies the property in* $\mathsf{SFR}(X)$.

(e) *If* $\mathbf{D}$ *is an ordered database schema and* X *is a finite set of order views with the maximal matching property, then if* X *satisfies the property in* $\mathsf{SFR}(X)$, *it also satisfies every property in* $\mathsf{PD}(X)$.

PROOF: Consult 2.1.5, 2.2.3, 2.3.15, 3.1.5, 3.3.7, and 4.2.3. □

5.1.3 Conclusions What can we conclude from this study? Here are some of the key points.

1. In the general setting, there are two *basic* equivalence classes of decompositions forms of which are independent of one another: pairwise definability and acyclicity. In the setting of a simple universal relational schema, these two form happen to coincide. However, adding even so little additional structure as a few functional dependencies is enough to separate the two. (See 3.2.2).

2. Acyclicity of the underlying hypergraph, by itself, does not seem to provide a useful characterization of desirable properties. The examples of 3.2 amply bear this out. Of course, it is possible that there is another, more useful, definition of the hypergraph associated with a set-based schema than that which we have developed. But that seems unlikely, as our construction is a natural one which is completely compatible with the universal relational notion.

3. Acyclicity of the underlying hypergraph is nonetheless extremely significant. When combined with pairwise definability to yield the *compound* class PDA, we obtain a new and highly nontrivial class of decompositions.

4. The generalization of the relational notion of a full reducer, by its very nature, requires additional (order) structure on the schema and views. When we add this structure, a generalization of the notion of full reducer to the set-based context is possible, and this generalization lies properly between pairwise definability alone and pairwise definability combined with acyclicity. However, we have not identified any other characterizations of decompositions which lie in this same class.

5.2 Further Directions

There are two further directions in which we intend to take the work established in this paper.

Application to a specific data model This paper establishes the mathematical results which tell us that the desirability characterizations of the universal relational model which are identified in 0.1.1 may be generalized to apply to a much wider class of database decompositions. However, it tells us nothing of the specific conclusions which may be obtained when we examine the meaning of these abstract characterizations within the context of a specific data model. In other words, we need to determine just how desirable these properties really are in specific contexts. Therefore, our next step will be to select a timely data model (probably an object-oriented model), and examine in detail the meaning of these various characterizations. Only through such a detailed study can we determine the extent to which these abstract results will have any practical use.

Consideration of stronger forms of acyclicity In [9], Fagin has shown that by considering only special types of acyclicity of universal relational schema hypergraphs, additional results may be obtained. In view of our conclusion above that acyclicity by itself is not a interesting characterization of desirability in the general context, one might ask if such an investigation would be of any practical importance. Our answer is that characterizations which require, say, both pairwise definability and a stronger form of acyclicity (such as the γ-acyclicity of [9]), might be extremely interesting, since γ-acyclicity has been shown to have important implications in the relational setting beyond those of ordinary acyclicity.

Acknowledgments

Initial work on the research reported herein was performed while the author was visiting the Department of Mathematics of the University of Oslo, Norway. He wishes to thank in particular the members of the Computational Linguistics Group for their kind hospitality during his stay there.

The author owes a special debt of gratitude to Ron Fagin, who gave the initial version of this paper a thorough reading and made numerous suggestions to improve the presentation.

References

[1] S. Alagić, *Object-Oriented Database Programming*, Springer-Verlag, 1989.

[2] F. Bancilhon and N. Spyratos, Independent components of databases, in *Proceedings of the Seventh International Conference on Very Large Data Bases*, pp. 398–408, 1981.

[3] F. Bancilhon and N. Spyratos, Update semantics of relational views, *ACM Trans. Database Systems*, **6**(1981), 557–575.

[4] C. Beeri, R. Fagin, D. Maier, and M. Yannakakis, On the desirability of acyclic database schemes, *J. Assoc. Comp. Mach.*, **30**(1983), 479–513.

[5] C. Berge, *Graphs and Hypergraphs*, North Holland, 1973.

[6] M. A. Casanova, R. Fagin, and C. H. Papadimitriou, Inclusion dependencies and their interaction with functional dependencies, *J. Comput. System Sci.*, **28**(1984), 29–59.

[7] S. Ceri, G. Gottlob, and L. Tanca, *Logic Programming and Databases*, Springer-Verlag, 1990.

[8] R. Fagin, Horn clauses and database dependencies, *J. Assoc. Comp. Mach.*, **29**(1982), 952–985.

[9] R. Fagin, Degrees of acyclicity for hypergraphs and relational database schemes, *J. Assoc. Comp. Mach.*, **30**(1983), 514–550.

[10] R. Fagin, A. O. Mendelzon, and J. D. Ullman, A simplified universal relation assumption and its properties, *ACM Trans. Database Systems*, **7**(1982), 343–360.

[11] R. Fagin and M. Y. Vardi, The theory of data dependencies – a survey, in *Mathematics of Information Processing* (M. Anshel and W. Gewirtz, eds.), pp. 19–71, American Mathematical Society, 1986.

[12] I. Fleischer, A note on subdirect products, *Acta Math. Acad. Sci. Hungar.*, **6**(1955), 463–465.

[13] H. Gallaire, J. Minker, and J. Nicolas, Logic and databases: a deductive approach, *ACM Comput. Surveys*, **16**(1984), 153–185.

[14] J. Grant and B. E. Jacobs, On the family of generalized dependency constraints, *J. Assoc. Comp. Mach.*, **29**(1982), 986–997.

[15] G. Grätzer, *Universal Algebra*, D. Van Nostrand, 1968.

[16] G. Grätzer, *General Lattice Theory*, Academic Press, 1978.

[17] S. J. Hegner, Unique complements and decompositions of database schemata, Technical Report PC 12/ 12.89, Centro di Ricerche in Fisica e Matematica (CERFIM), Locarno, Switzerland, 1989, Revised version to appear in *J. Comput. System Sci.*

[18] S. J. Hegner, Foundations of canonical update support for closed database views, in *ICDT'90, Third International Conference on Database Theory, Paris, France, December 1990* (S. Abiteboul and P. C. Kanellakis, eds.), pp. 422–436, Springer-Verlag, 1990.

[19] S. J. Hegner, Pairwise-definable subdirect decompositions of general database schemata, in *MFDBS91, Third Symposium on Mathematical Fundamentals of Database and Knowledge Base Systems, Rostock, Germany, May 1991* (B. Thalheim, J. Demetrovics, and H. Gerhardt, eds.), pp. 243–257, Springer-Verlag, 1991.

[20] H. Herrlich and G. E. Strecker, *Category Theory*, Allyn and Bacon, 1973.

[21] R. Hull and R. King, Semantic database modeling: survey, applications, and research issues, *ACM Comput. Surveys*, **19**(1987), 201–260.

[22] B. E. Jacobs, A. R. Aronson, and A. C. Klug, On interpretations of relational languages and solutions to the implied constraint problem, *ACM Trans. Database Systems*, **7**(1982), 291–315.

[23] D. Maier, *The Theory of Relational Databases*, Computer Science Press, 1983.

[24] D. Maier, Y. Sagiv, and M. Yannakakis, On the complexity of testing implications of functional and join dependencies, *J. Assoc. Comp. Mach.*, **28**(1981), 680–695.

[25] J. D. Monk, *Mathematical Logic*, Springer-Verlag, 1976.

[26] O. Ore, Theory of equivalence relations, *Duke Math. J*, **9**(1942), 573–627.

[27] J. Paredaens, The interaction of integrity constraints in an information system, *J. Comput. System Sci.*, **20**(1978), 310–329.

[28] J. Paredaens, P. De Bra, M. Gyssens, and D. Van Gucht, *The Structure of the Relational Database Model*, Springer-Verlag, 1989.

[29] R. Reiter, Towards a logical reconstruction of relational database theory, in *On Conceptual Modelling* (M. L. Brodie, J. Mylopoulos, and J. W. Schmidt, eds.), pp. 191–233, Springer-Verlag, 1984.

[30] F. Sadri and J. D. Ullman, Template dependencies: a large class of dependencies in relational databases, *J. Assoc. Comp. Mach.*, **29**(1982), 363–372.

[31] M. Y. Vardi, On decompositions of relational databases, in *Proceedings 23rd Annual Symposium on Foundations of Computer Science*, pp. 176–185, 1982.

[32] G. H. Wenzel, Note on a subdirect representation of universal algebras, *Acta Math. Acad. Sci. Hungar.*, **18**(1967), 329–333.

[33] O. Zariski and P. Samuel, *Commutative Algebra, Volume I*, Springer-Verlag, 1958.