Pairwise-Definable Subdirect Decompositions of General Database Schemata[†]

Stephen J. Hegner
Department of Computer Science and Electrical Engineering
Votey Building
University of Vermont
Burlington, Vermont 05405 USA

(802)656-3330
Internet: hegner@uvm.edu
USEnet: ..uunet!uvm-gen!hegner

This paper appeared in *Proceedings of the Third Symposium on Mathematical Fundamentals of Database and Knowledge Base Systems, 5-9 May 1991, Rostock, Germany* Springer-Verlag Lecture Notes in Computer Science Vol. 495, 1991, pp. 243-257.

Abstract

One of the most important results in the theory of decomposition of universal relational schemata is the equivalence of acyclicity of the hypergraph of the schema to numerous desirable properties regarding simplicity of constraints, correctness of query evaluation algorithms, and complexity of integrity maintenance. In this paper, we show that the thrust of these results is not specific to the relational model, but rather applies in a much more general context in which schemata are just sets and views are defined by surjective functions. This is accomplished by replacing the notion of hypergraph of a schema (which is specific to the relational model) with the much more general notion of pairwise definability, which is meaningful in the context of any decomposition into a set of views.

[†]Much of the research reported herein was performed while the author was visiting the Mathematics Department 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.

0. Introduction

A cornerstone of the theory of universal relational database schemata is that of *acyclic decompositions.* In that work, 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, are all shown to be equivalent to the property that the underlying hypergraph of the schema be acyclic [FMU82],[BFMY83]. With the growing importance of new data models, such as object-oriented models [Ala89], as well as extensions of the relational model to contexts including deduction [CGT90] and incomplete information [GMN84],[Rei84], it is important to have an understanding of the extent to which these results may be extended to more general classes of database schemata. The purpose of this paper is to initiate an investigation which will hopefully provide the foundation for such an understanding.

Our starting point is the main characterization theorem for acyclic schemata, as developed in [BFMY83, Thm. 3.4]. For reference, we reproduce that result below.

0.1 Theorem – the classical characterization of acyclicity *The following properties are equivalent on a universal relational schema* $\mathbf{R} = (R[U], \{\bowtie [U_1, U_2, .., U_n]\})$ *with single relation symbol* R *over attribute set* U*, governed by the single full join dependency* $\bowtie [U_1, U_2, .., U_n]$*, and viewed as being decomposed into the set of projections* $\{R_{U_1}[U_1], \ldots, R_{U_n}[U_n]\}$.

- (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 join expression.*
- (S6) *The join dependency* $\bowtie [U_1, U_2, .., U_n]$ *is equivalent to a set of multivalued dependencies.*
- (S7) *The join dependency* $\bowtie [U_1, U_2, .., U_n]$ *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* $\mathbf{R}$. □

These twelve equivalent properties may be partitioned into two groups. Items (S1)–(S8) deal with properties of the schema which are useful in an application sense. They tell us *why* acyclic decompositions are desirable. Properties (S9)–(S12), on the other hand, serve primarily as a characterization. They tell us *how* to determine whether or not a schema has properties (S1)–(S8).

The principal result of this paper states that properties (S1)–(S8), suitably recast, hold in a much more general setting. More specifically, we work within the most general possible framework — database schema are just sets and database mappings are just functions. Within this framework, we identify new properties (T1)–(T7) of decompositions, and establish that they have the following characteristics.

- For each i, $1 \leq i \leq 7$, (Ti) reduces to (Si) when specialized to traditional relational decompositions.
- The properties (T1)–(T7) are all equivalent to one another.

In addition, we may include a property (T8) which reduces to (S8) for traditional relational decompositions, provided we add somewhat more structure to our general framework. (We must work within a framework which supports the concept of database size to generalize the notion of full reducer.)

In short, our result establishes that the "desirable" properties of acyclic schemata are not special properties which make sense only in a relational setting; rather, they are very general properties which make sense in any reasonable database setting. On the other hand, we offer no generalizations of (S9)–(S12). This is to be expected, since these characterizations deal primarily with a structure, particular to the relational model, which permit us to characterize algorithmically when a schema has properties (S1)–(S8). We cannot expect such algorithms to extend to a much less structured framework.

Throughout this paper, we assume familiarity with the standard notation and terminology of the relational model, as may be found in [Mai83] and [PDGV89]. Specific familiarity with the theory of acyclicity in the relational model ([FMU82],[BFMY83] or [Mai83, Ch. 13]) will prove very helpful in understanding the generalizations, but is not absolutely necessary, since our discussion includes reviews of the key ideas.

In many cases, the proofs of the results are presented here are rather involved, and space limitations have forced us to omit them, giving this paper the form of an extended abstract. A full paper with all proofs will be available.

1. Schemata and Decompositions

In this section, we formulate the key definitions for the set-based framework in which our generalization of Theorem 0.1 will live. Many of these concepts are developed in more detail in [Heg89] and [Heg90] — our emphasis here is on how they generalize the traditional relational framework.

1.1 Universal relational schemata and projective views We begin by briefly recalling the relational framework in which the traditional acyclicity theory is presented, recast in a way compatible with our more general framework. By a *universal relational schema* we mean a pair $\mathbf{R} = (R[U], \Phi)$ in which R is a relation symbol, U is a finite set of attributes, and Φ is a set of implicational dependencies [Fag82],[FV86]. The set of all relations which satisfy each of the constraints in Φ (the *legal databases*) of $\mathbf{R}$ is denoted by $\mathsf{LDB}(\mathbf{R})$. If Ψ is another set of constraints, we say that Φ and Ψ are *logically equivalent* if $\mathsf{LDB}((R[U], \Phi)) = \mathsf{LDB}((R[U], \Psi))$.

In this work, the most important types of dependencies are join dependencies and multivalued dependencies, with which we assume familiarity. We shall always regard a multivalued dependency as a join dependency on two projections; for our purposes, the multivalued dependency $W \twoheadrightarrow V$ (with $W \cap V = \emptyset$) on attribute set U *is* the join dependency $\bowtie [W \cup V, U \setminus V]$.

The schema $(R[U], \{\bowtie [U_1, \ldots, U_n]\})$ with $U = \cup_{i=1}^{n} U_i$ is called a *simple universal relational schema*, and is often denoted, for emphasis, by $\mathbf{R}[U_1, \ldots, U_n]$.

For $W \subseteq U$, the W*-projection view* of $\mathbf{R}$ is the pair $\Pi_W = (\mathbf{R}_W, \pi_W)$. $\mathbf{R}_W$ is the universal relational schema $(R_W[W], \pi_W(\Phi))$ in which R_W is a relation symbol on the attributes W. $\pi_W : \mathbf{R} \to \mathbf{R}_W$ is the relational calculus query defining the projection onto the attributes of W. ${\pi_W}' : \mathsf{LDB}(\mathbf{R}) \to \mathsf{LDB}(\mathbf{R}_W)$ denotes the underlying function which sends each relation over U to its projection on the attributes of W. Finally, $\pi_W(\Phi)$ is the projection of the family Φ of implicational dependencies onto the attributes W; that is, the set of all dependencies which are satisfied by every member of $\mathsf{LDB}(\mathbf{R}_W)$. It is well known that such a projection of implicational dependencies itself has a basis consisting of implicational dependencies [Fag82, 6.3], so that ${\pi_W}'$ is surjective. In the case that $\Phi = \{\bowtie [U_1, \ldots, U_n]\}$ and $W \subseteq U_i$ for some i, $\pi(\Phi)$ contains no nontrivial dependencies — that is, there are no constraints on $\mathsf{LDB}(\mathbf{R}_W)$.

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 $\emptyset$ (which is always the case in nontrivial examples), $\mathsf{LDB}(\mathbf{R}_\emptyset)$ 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.

The traditional acyclicity theory is usually applied to the simple schema $\mathbf{R}[U_1, \ldots, U_n]$, which is often denoted in the literature by just $\{U_1, \ldots, U_n\}$, and the decomposition is actually into the projective views $\{\Pi_{U_1}, \ldots \Pi_{U_n}\}$.

1.2 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 $R[U]$ 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* $\mathbf{D}$ is entirely defined by a set of *legal databases* (or *legal states*), which we denote by $\mathsf{LDB}(\mathbf{D})$. A *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

$\mathbf{V}$ is a set-based database schema and $\gamma : \mathbf{D} \to \mathbf{V}$ is a set-based morphism with the property that $\gamma' : \mathsf{LDB}(\mathbf{D}) \to \mathsf{LDB}(\mathbf{V})$ is surjective. The set of all views of $\mathbf{D}$ is denoted $\mathsf{View}(\mathbf{D})$.

The *congruence* $\mathsf{Congr}(\Gamma)$ of Γ (denoted $\equiv_\Gamma$ in [BS81]) is the equivalence relation on $\mathsf{LDB}(\mathbf{D})$ defined by $(M_1, M_2) \in \mathsf{Congr}(\Gamma)$ iff $\gamma'(M_1) = \gamma'(M_2)$. Given set-based views $\Gamma_1 = (\mathbf{V}_{\Gamma_1}, \mu_{\Gamma_1})$ and $\Gamma_2 = (\mathbf{V}_{\Gamma_2}, \mu_{\Gamma_2})$, a *view morphism* $f : \Gamma_1 \to \Gamma_2$ is a set-based database morphism $f : \mathbf{V}_1 \to \mathbf{V}_2$ such that the following diagram commutes.

$$\begin{tikzcd} & \mathbf{D} \arrow[dl, "\gamma_1"'] \arrow[dr, "\gamma_2"] & \\ \mathbf{V}_1 \arrow[rr, dashed, "f"] & & \mathbf{V}_2 \end{tikzcd}$$

It is easy to show [Heg90, 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 furthermore allows us to regard Γ_2 as a view of $\mathbf{V}_1$ when $\Gamma_2 \leq \Gamma_1$. We call this new view of $\mathbf{V}_1$ the *relativization* of Γ_2 to Γ_1 and denote it by $\Lambda(\Gamma_1, \Gamma_2) = (\mathbf{V}_2, \lambda(\Gamma_1, \Gamma_2))$. The relativization is relational if Γ_1 and Γ_2 are [Heg90, 3.4].

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 [HS73, 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 $\Gamma_2 \leq \Gamma_1$.

1.3 Example To make sure that there is no confusion, we illustrate these ideas with a simple relational example. Let $\mathbf{E} = (R[ABC], \{\bowtie [AB, BC], B \to C\})$. $\mathsf{LDB}(\mathbf{E})$ is just the set of all databases which satisfy the join dependency $\bowtie [AB, BC]$. 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 $\Pi_B \leq \Pi_{AB}$. 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}.

1.4 The decomposition morphism of a set of views In the simple relational case, the *decomposition mapping* $\Delta\langle U_1, \ldots, U_n\rangle : \mathbf{R}[U_1, \ldots, U_n] \to \mathbf{V}_1 \times \ldots \times \mathbf{V}_n$ is defined on elements by $M \mapsto (\pi_{U_1}{}'(M), \ldots, \pi_{U_n}{}'(M))$. We think of this mapping as defined by the set of *views* $\{\Pi_{U_1}, \ldots, \Pi_{U_n}\}$, and generalize this concept to the set-based case as follows. Given a schema $\mathbf{D}$ and a set $X = \{\Gamma_1, \ldots, \Gamma_n\}$ of views with $\Gamma_i = (\mathbf{V}_{\Gamma_i}, \mu_{\Gamma_i})$, the decomposition morphism $\Delta\langle X\rangle : \mathbf{D} \to \mathbf{V}_1 \times \ldots \times \mathbf{V}_n$ is given on elements by $M \mapsto (\gamma_1{}'(M), \ldots, \gamma_n{}'(M))$. We call X a *subdirect decomposition*[1] 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 *decomposed database*; a subdirect decomposition then is precisely one in 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 will thus make use of properties of subdirect decompositions, rather than properties of join dependencies. To this end, given a join dependency $\bowtie[U_1, \ldots, U_n]$, we define the *decomposition of* $\bowtie[U_1, \ldots, U_n]$ to be the set of views $\{\Pi_{U_1}, \ldots, \Pi_{U_n}\}$. It is immediate that if $\Phi \models \bowtie[U_1, \ldots, U_n]$ for the schema $\mathbf{R} = (R[U], \Phi)$, then $\{\Pi_{U_1}, \ldots, \Pi_{U_n}\}$ is a subdirect decomposition of $\mathbf{R}$. Unfortunately, the converse is not always the case; there exists a decomposition into projections such that the join is not a reconstruction map. See [Var82, p. 183] for an example. However, such a counterexample requires that Φ contain embedded dependencies; when Φ consists entirely of total dependencies [Fag82], there is a natural bijective correspondence between subdirect decompositions into projections and join dependencies, as recaptured by the following.

1.5 Proposition *Let* $\mathbf{R}$ *be the universal relational schema* $(R[U], \Phi)$*, with* Φ *a set of total implicational dependencies, and suppose that* $U_1, \ldots, U_n \subseteq U$ *with* $\bigcup_{i=1}^{n} U_i = U$*. Then* $\{\Pi_{U_1}, \ldots, \Pi_{U_n}\}$ *is a subdirect decomposition of* $\mathbf{R}$ *iff* $\Phi \models \bowtie[U_1, \ldots, U_n]$*.*

PROOF: Consult [Var82, Cor. 4]. $\square$

Note in particular that the above result holds when $\Phi = \{\bowtie[U_1, \ldots, U_n]\}$, which is the context in which acyclicity theory is usually considered. Thus, for this most fundamental example, the context in which the classical acyclicity theory is usually formulated, subdirect decompositions generalize join-based decompositions without compromise.

1.6 Consistency of views In the context of a universal relational schema $\mathbf{R} = (R[U], \Phi)$, let $W_1, W_2 \subseteq U$. 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 $\lambda(\Pi_{W_1}, \Pi_{W_1 \cap W_2})'(M_1) = \lambda(\Pi_{W_2}, \Pi_{W_1 \cap W_2})'(M_2)$. 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 $U_i \cap U_j = \emptyset$, 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 $\Gamma_1 = (\mathbf{V}_{\Gamma_1}, \mu_{\Gamma_1})$, $\Gamma_2 = (\mathbf{V}_{\Gamma_2}, \mu_{\Gamma_2})$, and $\Gamma = (\mathbf{V}_\Gamma, \mu_\Gamma)$ be arbitrary views of the set-based schema $\mathbf{D}$ with the property that

[1]The terminology is borrowed from universal algebra; see [Gra68, §20] for details.

$\Gamma \leq \Gamma_1$ and $\Gamma \leq \Gamma_2$. We say that $M_1 \in \mathsf{LDB}(\mathbf{V}_1)$ and $M_2 \in \mathsf{LDB}(\mathbf{V}_2)$ 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 $\mathbf{R}$, $\Gamma_1 = \Pi_{W_1}$, 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. 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 *fully commuting pair*. In the case that $\{\Gamma_1, \Gamma_2\}$ is a subdirect decomposition of $\mathbf{D}$ in addition to being a fully commuting pair, we call it a *fully commuting complementary pair*. In the case that the constraints other than the governing join dependency embed into the projections, the relational notion of consistency reduces to commuting congruences, as recorded formally in the following proposition.

1.7 Proposition *For the schema* $\mathbf{R} = (R[U], \Phi \cup \{\bowtie[U_1,\ldots,U_n]\})$*, every pair of views in* $\{\Pi_{U_1},\ldots,\Pi_{U_n}\}$ *has commuting congruences with* $\mathsf{Congr}(\Pi_{U_i}) \circ \mathsf{Congr}(\Pi_{U_j}) = \mathsf{Congr}(\Pi_{U_i \cap U_j})$ *iff* $\{\pi_{U_i}{}'(\Phi) \mid 1 \leq i \leq n\} \cup \{\bowtie[U_1,\ldots,U_n]\} \models \Phi$. *(We regard each* $\varphi \in \pi_{U_i}{}'(\Phi)$ *as an embedded constraint; by definition,* $M \in \mathsf{LDB}(\mathbf{R})$ *satisfies* φ *iff* $\pi_{U_i}{}'(M)$ *does.)* $\square$

This result is of fundamental importance in establishing the various equivalent notions of schema simplicity. It is also very useful in other contexts. For example, in [Heg90], we show that the notion of commuting congruences is of fundamental importance in 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 the notion 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 [Fle55] and [Wen67] for details.

1.8 The bounded weak partial lattice of views In the set-based context, the set $[\mathsf{View}(\mathbf{D})]$ of all equivalence classes of views (equivalent views having the same congruence) has a natural lattice-like structure which will prove useful. This structure is compatible with the ordering $\leq$ defined in 1.2, but is not completely defined by it. Specifically, for any two equivalence classes of views $[\Gamma_1]$ and $[\Gamma_2]$, we define their *join* to be the equivalence class of views whose congruence is $\mathsf{Congr}(\Gamma_1) \cap \mathsf{Congr}(\Gamma_2)$, and denote this class by $[\Gamma_1] \vee [\Gamma_2]$. As an abuse of notation, we shall often write just $\Gamma_1 \vee \Gamma_2$, with the understanding that this notation identifies some canonical representative. This operation is clearly associative, and we write $\bigvee_{i=1}^{n} \Gamma_i$ for $\Gamma_1 \vee \ldots \vee \Gamma_n$. Note that the join operation is none other than the supremum operation associated with the order $\leq$. Remarkably, in the universal relational context of $\mathbf{R}[U_1,\ldots,U_n]$, we actually have that $\Pi_{U_i} \vee \Pi_{U_j}$ is isomorphic to the view which computes the relational join $M \mapsto \pi_{U_i}{}'(M) \bowtie \pi_{U_j}{}'$. However, our terminology is motivated from lattice-theoretic considerations, and the identity of terminology is something of a coincidence.

We also have a meet operation, but it must be defined more carefully. Specifically, for any views Γ_1 and Γ_2, we define their *meet* as follows.

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

As with the case of join, we will often write $\Gamma_1 \wedge \Gamma_2$ to denote a canonical representative of $[\Gamma_1] \wedge [\Gamma_2]$. However, unlike the join, the meet is a partial operation, and not simply the infimum under $\leq$. Under the ordering $\leq$, while the infimum of a pair of views always exists, their meet is only defined if the congruences of those views commute.

There are two special views in $[\mathsf{View}(\mathbf{D})]$. 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 its canonical representative is denoted $\Gamma_\perp(\mathbf{D})$. Note that $\Gamma_\perp(\mathbf{D}) \leq \Gamma \leq \Gamma_\top(\mathbf{D})$ for any view $\mathbf{D}$.

The structure $[\mathbf{View}(\mathbf{D})] = ([\mathbf{View}(\mathbf{D})], \vee, \wedge, [\Gamma_\top(\mathbf{D})], [\Gamma_\perp(\mathbf{D})])$ is called the *view algebra* of $\mathbf{D}$. It is a *bounded weak partial lattice*, in the sense of [Gra78, pp. 40-44]. It is similar to a bounded lattice (with greatest element $[\Gamma_\top(\mathbf{D})]$ and least element $[\Gamma_\perp(\mathbf{D})]$), the difference being that meet is only a partial operation.

Given a subset $X \subseteq [\mathsf{View}(\mathbf{D})]$, $\mathsf{Span}(X)$ denotes the closure of X under the operations of join and meet. In other words, it is the smallest set of views such that $X \subseteq \mathsf{Span}(X)$, and $\Gamma_1, \Gamma_2 \in \mathsf{Span}(X)$ implies $\Gamma_1 \vee \Gamma_2 \in \mathsf{Span}(X)$, and if it is defined, $\Gamma_1 \wedge \Gamma_2 \in \mathsf{Span}(X)$ also. We sometimes call $\mathsf{Span}(X)$ the *partial sublattice* generated by X. Note, however, that it need not have a greatest or least element. There are two important characterizations which may be expressed in terms of $[\mathbf{View}(\mathbf{D})]$.

1.9 Proposition — characterization of subdirect decomposition *Let $\mathbf{D}$ be a set-based schema, and let $X = \{\Gamma_1, \ldots, \Gamma_n\}$ be a set of views of $\mathbf{D}$ with $\Gamma_i = (\mathbf{V}_{\Gamma_i}, \mu_{\Gamma_i})$. Then X is a subdirect decomposition of $\mathbf{D}$ iff $\bigvee_{i \in I} \Gamma_i = \Gamma_\top(\mathbf{D})$; that is, iff $\bigcap_{i \in I} \mathsf{Congr}(\Gamma_i) = \mathbf{1}_{\mathsf{LDB}(\mathbf{D})}$, the identity relation on $\mathsf{LDB}(\mathbf{D})$.*

PROOF OUTLINE: This is essentially a restatement of a well-known theorem of universal algebra; see, *e.g.*, [Gra68, §20, Thm. 2]. $\square$

1.10 Consistency dependencies The notion of consistency gives rise to an important class of dependencies which are always satisfied by decomposed databases. In the context of the simple universal relational schema $\mathbf{R}[U_1, U_2, \ldots, U_n]$, the tuple of instances $(M_1, \ldots, M_n) \in \Delta\langle U_1, \ldots, U_n\rangle'(\mathsf{LDB}(\mathbf{R}))$ satisfies the $[U_i, U_j]$-*consistency dependency* if $\lambda(\Pi_{U_i}, \Pi_{U_i \cap U_j})'(M_i)$ $\lambda(\Pi_{U_j}, \Pi_{U_i \cap U_j})'(M_j)$. We denote this dependency by $\asymp[U_i, U_j]$, and write $(M_1, \ldots, M_n) \models \asymp[U_i, U_j]$ to denote that $(M_1, \ldots, M_n)$ satisfies it.

Generalizing to the set-based case, let $X = \{\Gamma_1, \ldots, \Gamma_n\}$ be a set of views of the schema $\mathbf{D}$, with $\Gamma_i = (\mathbf{V}_{\Gamma_i}, \mu_{\Gamma_i})$ for $1 \leq i \leq n$. The tuple of instances $(M_1, \ldots, M_n) \in \mathsf{LDB}(\mathbf{V}_1) \times \ldots \times \mathsf{LDB}(\mathbf{V}_n)$ satisfies the $[\Gamma_i, \Gamma_j]$-*consistency dependency* for X if $\Gamma_i \wedge \Gamma_j$ exists, and that, whenever $(M_1, \ldots, M_n) \in \mathsf{LDB}(\mathbf{V}_1) \times \ldots \times \mathsf{LDB}(\mathbf{V}_n)$, the condition $\lambda(\Gamma_i, \Gamma_i \wedge \Gamma_j)'(M_i) = \lambda(\Gamma_j, \Gamma_i \wedge \Gamma j)'(M_j)$ holds. We denote this dependency by $\asymp[\Gamma_i, \Gamma_j]$, and write $(M_1, \ldots, M_n) \models \asymp[\Gamma_i, \Gamma_j]$

to denote that $(M_1, \ldots, M_n)$ satisfies this dependency. It is immediate that, on $\mathbf{R}[U_1, \ldots, U_n]$, $\asymp[\Pi_{U_i}, \Pi_{U_j}]$ is $\asymp[U_i, U_j]$.

2. The Principal Characterizations

In this section, we elaborate the generalizations, to the set-based framework, of the properties (S1)–(S7) of Theorem A.

2.1 The Context Throughout this section, we let $\mathbf{D}$ be a set-based schema, with $X = \{\Gamma_1, .., \Gamma_n\}$ a subdirect decomposition of $\mathbf{D}$ and $\Gamma_i = (\mathbf{V}_{\Gamma_i}, \mu_{\Gamma_i})$. Unless specifically stated otherwise, we also let $\mathbf{R}$ denote the simple universal relational schema $\mathbf{R}[U_1, U_2, .., U_n]$. When we apply general set-based decomposition theory to the simple relational schema $\mathbf{R}[U_1, \ldots, U_n]$, we take the set of views X to be $\{\Pi_{U_1}, \ldots, \Pi_{U_n}\}$.

The Principal Characterization

In the classical theory on simple universal relational schemata, the "defining" notion is that of acyclicity of the underlying hypergraph. Since it is not at all clear how to generalize the notion of underlying hypergraph to the general set-based context, in our more general context, we seek an alternate defining notion. The one we provide, called pairwise definability, recaptures what appears to be a very basic notion of simplicity of a decomposition.

2.2 Pairwise definability The business of characterizing subdirect decompositions amounts to identifying the types of constraints which are necessary to describe the set of decomposed databases. In general, to determine whether or not $(M_1, \ldots, M_n) \in \mathsf{LDB}(\mathbf{V}_1) \times \ldots \times \mathsf{LDB}(\mathbf{V}_n)$ is a decomposed database, we must examine the entire n-tuple $(M_1, \ldots, M_n)$ globally, as a single entity. However, in some cases, the checking can be much more local, considering only certain pairs of the form (M_i, M_j). It is this sort of localization of constraints on the decomposed schema which provides the cornerstone for our generalized notion of simplicity. Specifically, the set of set-based views X is *pairwise definable* if there is a set Y of pairs of elements of X such that $\Delta\langle X\rangle'(\mathsf{LDB}(\mathbf{D})) = \{(M_1, \ldots, M_n) \in \mathsf{LDB}(\mathbf{V}_1) \times \ldots \times \mathsf{LDB}(\mathbf{V}_n) \mid (\forall\{\Gamma_i, \Gamma_j\} \in Y)((M_1, \ldots, M_n) \models \asymp[\Gamma_i, \Gamma_j]\}$. In words, the decomposition of $\mathbf{D}$ defined by the set of views X is pairwise definable if all of the constraints on the decomposed database are of two forms:

(i) The local constraints on each $\mathsf{LDB}(\mathbf{V}_i)$.

(ii) A set of constraints which state the certain pairs of the views are consistent, in that they agree on their common data.

In the simple relational case, each $Y \in X$ is of the form $\{\Pi_{U_i}, \Pi_{U_j}\}$, and pairwise definability asserts that that it suffices to check for consistency on the components defined by these pairs of projections.

Although it is not usually identified as an equivalent characterization of acyclicity in the relational case, it is in fact so, and we record this fact formally.

2.3 Proposition *The simple universal relational schema* $\mathbf{R}[U_1, \ldots, U_n]$ *is acyclic iff it is pairwise definable.* $\square$

Even in the simple relational case, we feel that pairwise definability provides a much more natural characterization of "desirable" decompositions than does hypergraph acyclicity. We now proceed to establish generalizations of each of the conditions (S1)-(S7) of Theorem 0.1, each equivalent to pairwise definability.

Generalization of Condition (S1)

2.4 Pairwise and total consistency In the context of the simple universal relational schema $\mathbf{R}[U_1, \ldots, U_n]$, the tuple of instances $(M_1, \ldots, M_n) \in \mathsf{LDB}(\mathbf{R}_{U_1}) \times \ldots \times \mathsf{LDB}(\mathbf{R}_{U_n})$ is *pairwise consistent* if, for every pair $\{U_i, U_j\}$, we have $\lambda(\Pi_{U_i}, \Pi_{U_i \cap U_j})'(M_i) = \lambda(\Pi_{U_j}, \Pi_{U_i \cap U_j})'(M_j)$. $(M_1, \ldots, M_n)$ is *globally consistent* if it is a decomposed database; *i.e.*, if it is in $\Delta\langle U_1, \ldots, U_n\rangle'(\mathsf{LDB}(\mathbf{R}))$. Condition (S1) states that $\bowtie [U_1, \ldots, U_n]$ is acyclic iff every pairwise consistent database over $\mathbf{R}[U_1, \ldots, U_n]$ is globally consistent. Note that pairwise consistency is apparently a slightly stronger condition than pairwise definability, since in the former, we require that all pairs of component views be consistent, while in the latter we require only that sufficiently many be consistent. Of course, the two conditions turn out to be equivalent, since they both characterize acyclicity.

Using the ideas developed in the previous section, this is easy to generalize to the set-based context. We say that the tuple of instances $(M_1, \ldots, M_n) \in \mathsf{LDB}(\mathbf{V}_1) \times \ldots \times \mathsf{LDB}(\mathbf{V}_n)$ is *pairwise consistent* on X if, for every pair $\{\Gamma_i, \Gamma_j\} \in X$, if Γ_i and Γ_j have commuting congruences, then $\lambda(\Gamma_i, \Gamma_i \wedge \Gamma_j)'(M_i) = \lambda(\Gamma_j, \Gamma_i \wedge \Gamma j)'(M_j)$. In view of Proposition 1.7, it is clear that this definition generalizes the one above for the universal relational context. As in the relational case, $(M_1, \ldots, M_n)$ is *globally consistent* on X if it is a decomposed database; *i.e.*, if it is in $\Delta\langle X\rangle'(\mathsf{LDB}(\mathbf{D}))$. We define the generalization (T1) of (S1) as follows.

(T1) Every pairwise consistent tuple of instances $(M_1, \ldots, M_n)$ on X is globally consistent.

Generalization of Condition (S2)

2.5 Compatibility trees For the universal relational schema $\mathbf{R}[U_1, \ldots, U_n]$, the *complete intersection graph* is the undirected graph which has as nodes the members of $\{U_1, U_2, .., U_n\}$, with the edge from U_i to U_j labelled by $U_i \cap U_j$. An *intersection graph* for $\mathbf{R}$ is any subgraph of the complete intersection graph which is obtained by deleting only edges, and not nodes. A *join graph* is an intersection graph with the property that for every i and j, there is a path from U_i to U_j which contains every attribute common to U_i and U_j in every edge label on that path. A *join tree* is a join graph which is a tree.

To generalize this idea to the set-based context, we employ the notion, established in Proposition 1.7, that view meet generalizes attribute set intersection. 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 $\{\Gamma_i \wedge \Gamma_j \mid \Gamma_i, \Gamma_j \in X\}$.

There is a labelled edge from Γ_i to Γ_j precisely in the case that $\{\Gamma_i, \Gamma_j\}$ is a fully commuting pair. In this case, the label is $\Gamma_i \wedge \Gamma_j$. 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 $\Gamma_i, \Gamma_j \in X$, a Γ-*path* from Γ_i to Γ_j is a path from Γ_i to Γ_j 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 $\Gamma_i, \Gamma_j \in X$ and every view Γ with the property that $\Gamma \leq \Gamma_i$ and $\Gamma \leq \Gamma_j$, there is a Γ-path from Γ_i to Γ_j. A *compatibility tree* is a compatibility graph which is a tree.

It is immediate that the notions of complete meet graph, meet graph, compatibility graph, and compatibility tree reduce to complete intersection graph, intersection graph, join graph, and join tree, respectively, in the the setting of a simple universal relational schemata. In particular, existence of a compatibility tree, when restricted to the traditional relational case, reduces to existence of a join tree. We thus have the following generalization (T2) of (S2).

(T2) X has a compatibility tree.

Generalization of Condition (S3)

2.6 The running meet property The universal relational schema $\mathbf{R}[U_1, \ldots, U_n]$ has the *running intersection* property if there is a permutation σ of $\{1, .., n\}$ such that for every i, $1 < \sigma(i) \leq n$, there is a j with $\sigma(j) < \sigma(i)$ such that $(\bigcup_{k=1}^{i-1} U_{\sigma(k)}) \cap U_{\sigma(i)} \subseteq U_{\sigma(j)}$. This generalizes to the set-based case by defining the set of views X of the schema $\mathbf{D}$ to have the *running meet* property if there is a permutation σ of $\{1, .., n\}$ such that for each i, $1 < \sigma(i) \leq n$, there is a j with $\sigma(j) < \sigma(i)$ such that $(\bigvee_{k=1}^{i-1} \Gamma_{\sigma(k)}) \wedge \Gamma_{\sigma(i)} \leq \Gamma_{\sigma(j)}$. In view of Proposition 1.7, it is immediate that the running meet property, when restricted to traditional relational views, yields precisely the running intersection property. Thus, we define (T3), the generalization of (S3), as follows.

(T3) X has the running meet property.

Generalization of Condition (S4)

2.7 Monotone and commuting join expressions In the context of the traditional relational schema $\mathbf{R}[U_1, \ldots, U_n]$, the *join expressions* are the smallest class closed under the following operations.

(i) Each U_i 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 an expression on an n-tuple $(M_1, \ldots, M_n) \in \mathsf{LDB}(\mathbf{R}_{U_1}) \times \ldots \times \mathsf{LDB}(\mathbf{R}_{U_n})$ is in the obvious manner — we just substitute M_i for U_i and evaluate. Such an expression is called *complete* if it is an inverse to the decomposition map $\Delta\langle U_1, \ldots, U_n\rangle'$, and it is called *monotone* if each pair of relations which is joined in the evaluation of any $(M_1, \ldots, M_n) \in \Delta\langle U_1, \ldots, U_n\rangle'(\mathsf{LDB}(\mathbf{D}))$ is consistent.

To convert these ideas to a more general setting requires a bit of work. First of all, let us identify an alternate representation. A *join plan* [Mai83, Def. 13.4] for the universal relational schema $\mathbf{R}[U_1, \ldots, U_n]$ is a nonempty binary tree which has the following properties.

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

(ii) The leaves are labelled with (some of) the elements of $\{U_1, U_2, .., U_n\}$.

(iii) Each nonleaf node is labelled with the union of the attributes labelling its two sons.

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 U_i) is just U_i. 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. We call the tree *complete* (resp. *monotone*) if its underlying join expression is.

The generalization to the set-based case proceeds as follows. A *generalized join plan* for X is a binary tree which has the following properties.

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

(ii) The leaves are labelled with the elements of X.

(iii) Each nonleaf node is labelled with the join ("$\vee$") of the labels of its two sons.

As in the relational case, we call a generalized join plan *complete* if it reconstructs the schema $\mathbf{D}$; *i.e.*, if the congruence of the view labelling the root is the identity on $\mathsf{LDB}(\mathbf{D})$. To directly generalize the notion of a monotone join plan, we would need to have available an order structure on the databases of each schema. Instead, we use a condition which does not require an order structure. Call a generalized join plan *fully commuting* if, for each non-leaf node of the tree, the views which labels its two sons form a fully commuting pair. In other words, we replace the order-based property of monotonicity with the more basic property of commuting congruences.

It is unfortunately not the case that monotonicity reduces exactly to full commutativity. As a specific example, consider the schema $\mathbf{R}[AB, BC, CD]$. Then the join plan corresponding to the expression $((AB \bowtie CD) \bowtie BC)$ is clearly not monotone, although it is fully commuting. The "problem" is that $\bowtie [AB, CD]$ is not an embedded dependency of the schema. If we only allow such cross dependencies on disjoint sets of attributes when they are necessary, we can repair the situation. Specifically, let us call the join plan T *normal* if, whenever W_1 and W_2 are the labels of two sons of a node k in T, then $W_1 \cap W_2 = \emptyset$ implies that $\bowtie [W_1, W_2] \models \bowtie [U_1, \ldots, U_n]$. In other words, if there is a join on disjoint attributes, then that join is "necessary" in the sense that it embeds into the governing join dependency. We then have the following.

2.8 Proposition

(a) *The relational schema* $\mathbf{R}[U_1, \ldots, U_n]$ *has a complete monotone join plan iff it has a complete fully commuting join plan.*

(b) *A complete normal join plan for the relational schema* $\mathbf{R}[U_1, \ldots, U_n]$ *is monotone iff it is fully commuting.* $\square$

We are thus justified in defining (T4), the generalization of (S4), as follows.

(T4) X admits a fully commuting complete generalized join plan.

Generalization of Condition (S5)

2.9 Sequential join expressions Condition (S5) generalizes (S4) slightly by requiring that the join expression (or, equivalently, join plan) be sequential. A join plan (resp. generalized join plan) is *sequential* if the right subtree of each nonleaf node is a leaf. In other words, in each step of the evaluation, one of the operands is a basic projection which has yet to be joined with anything else. We have the following generalization (T5) of (S5).

(T5) X admits a fully commuting complete sequential generalized join plan.

Generalization of Condition (S6)

Condition (S6) stipulates that the the governing join dependency of the decomposition be equivalent to a set of multivalued dependencies. The usual characterizations of logical equivalence of sets of dependencies are model theoretic. However, model theoretic notions are difficult to carry over directly to the set-based framework. Therefore, to accommodate our generalization, we present the following alternate characterization, in terms of the partial sublattice generated by each set of dependencies.

2.10 Proposition — algebraic characterization of the equivalence of a join dependency to a set of multivalued dependencies *Let* $\Psi = \{\bowtie[W_{i1}, W_{i2}] \mid 1 \leq i \leq k\}$ *be a set of multivalued dependencies on the attribute set* U *of the universal relational schema* $\mathbf{R}[U_1, \ldots, U_n]$. *Then* Ψ *is logically equivalent to* $\{\bowtie[U_1, \ldots, U_n]\}$ *iff* $\mathsf{Span}(\{\Pi_{W_{ij}} \mid 1 \leq i \leq k\ ,\ 1 \leq j \leq 2\}) = \mathsf{Span}(\{\Pi_{U_1} \mid 1 \leq i \leq n\})$. *In other words,* Ψ *and* $\{\bowtie[U_1, \ldots, U_n]\}$ *are logically equivalent iff they generate the same partial sublattice of* $[\mathsf{View}(\mathbf{R})]$. $\square$

2.11 Equivalence of decompositions Generalizing to the context of the set-based schema $\mathbf{D}$, we say that two sets of views X_1 and X_2 of $\mathbf{D}$ are *decomposition equivalent* if $\mathsf{Span}(X_1) = \mathsf{Span}(X_2)$. A set $\Psi = \{\{Y_{i1}, Y_{i2}\} \mid 1 \leq i \leq k\}$ of subdirect complementary pairs (generalizing a set of multivalued dependencies) is *decomposition equivalent* to a set X of views (generalizing a single join dependency) if $\mathsf{Span}(X) = \mathsf{Span}(\{Y_{ij} \mid 1 \leq i \leq k\ ,\ 1 \leq j \leq 2\})$. In light of the previous proposition, the following condition generalizes (S6).

(T6) There is a set Ψ of subdirect complementary pairs which is decomposition equivalent to X.

Generalization of condition (S7)

2.12 Conflict-free multivalued dependencies Condition (S7) extends (S6) by requiring that the governing join dependency be equivalent to a set of conflict-free multivalued dependencies. The definition of conflict-freedom of a set of multivalued dependencies, as given in the literature, is a rather technical one. (See, *e.g.*, [BFMY83, Sec. 8, Def. A].) Furthermore, it is a syntactic one, in that a set of multivalued dependencies which is conflict free may be logically equivalent to one which is not. (See, *e.g.*, [BFMY83, Example 8.2].) Therefore, we do not reproduce the original definition here, but rather we provide the following equivalent — and we feel much more intuitive — characterization.

2.13 Proposition — algebraic characterization of conflict-free sets of multivalued dependencies *Let Ψ be a set of multivalued dependencies over the set U of attributes. Then Ψ is equivalent to a conflict-free set of multivalued dependencies iff for any $\bowtie[W_1, W_2] \in \Psi$, $\{\Pi_{W_1}, \Pi_{W_2}\}$ is a meet complementary pair of views of the schema $(R[U], \Psi)$.* $\square$

If we combine the previous proposition to Proposition 1.7, we see that Ψ is conflict free iff for any $\psi \in \Psi$, the remaining dependencies in Ψ have an embedded cover in the decomposition. In other words, the decomposition does not "split" dependencies across components of the schema, except for the multivalued dependency used to do the decomposition. For another result relating commuting congruences to embedded covers, see [Heg90, 3.10].

2.14 Conflict-free meet-complementary decompositions Generalizing the relational notion in light of the above, we say that a family Ψ of subdirect complementary pairs of views of the set-based schema $\mathbf{D}$ is *conflict free* if any $\{\Omega_1, \Omega_2\} \in \Psi$ is fully commuting. The generalization of (S7) then becomes the following.

(T7) There is a set Ψ of meet complementary pairs which is decomposition equivalent to X.

The Generalized Characterization Theorem

We now are in a position to state the main theorem of this paper, the promised generalization of Theorem 0.1. In addition to listing the properties (T1)-(T7), we include our fundamental notion of pairwise definability as condition (T0).

2.15 The main theorem *Let $\mathbf{D}$ be a set-based database schema, and let X be a subdirect decomposition of $\mathbf{D}$. Then the following conditions are equivalent.*

(T0) *X is pairwise definable.*

(T1) *Every pairwise consistent tuple of instances $(M_1, \ldots, M_n)$ on X is globally consistent.*

(T2) *X has a compatibility tree.*

(T3) *X has the running meet property.*

(T4) *X admits a fully commuting complete generalized join plan.*

(T5) *X admits a fully commuting complete sequential generalized join plan.*

(T6) *There is a set Ψ of subdirect complementary pairs which is decomposition equivalent to X.*

(T7) *There is a set Ψ of meet complementary pairs which is decomposition equivalent to X.* $\square$

3. The Order-Based Characterization

In this section, we develop the context necessary to generalize the notion of a full reducer, namely schemata with certain order properties.

3.1 Full reducers in the relational case Suppose we are working with the simple universal relational schema $\mathbf{R}[U_1, \ldots, U_n]$, and we are given a state $\vec{M} = (M_1, \ldots, M_n) \in \mathsf{LDB}(\mathbf{R}_{U_1}) \times \ldots \times \mathsf{LDB}(\mathbf{R}_{U_n})$, but it is not necessarily the case that $\vec{M} \in \Delta\langle U_1, \ldots, U_n\rangle'(\mathsf{LDB}(\mathbf{R}[U_1, \ldots, U_n]))$. 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) $\vec{N} = (N_1, \ldots, N_n) \in \mathsf{LDB}(\mathbf{R}_{U_1}) \times \ldots \times \mathsf{LDB}(\mathbf{R}_{U_n})$ such that $\vec{N} \subseteq \vec{M}$ and $\vec{N} \in \Delta\langle U_1, \ldots, U_n\rangle'(\mathsf{LDB}(\mathbf{R})[U_1, \ldots, U_n])$. Indeed, $\vec{N}$ if found by simply joining the M_i's and discarding those tuples which do not join. We call $\vec{N}$ the *full reduction* of $\vec{M}$. A *sequential full reducer* (called just a *full reducer* in the literature) is a program which computes this $\vec{N}$ is a nice way. More precisely, the *semijoin* operation $U_i \ltimes U_j$ applied to a pair $(M_i, M_j) \in \mathsf{LDB}(\mathbf{R}_{U_i}) \times \mathsf{LDB}(\mathbf{R}_{U_j})$ yields $\pi_{U_i}{}'(M_i \bowtie M_j)$; this result is often denoted by $M_i \ltimes M_j$. A *(sequential) full reducer* is a sequence of replacement operations of the form $M_i \leftarrow M_i \ltimes M_j$. 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 $\mathbf{R}[U_1, \ldots, U_n]$ is acyclic.

3.2 Ordered schemata and morphisms 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 $\mathbf{V}_1 \times \ldots \times \mathbf{V}_n$ we use the product ordering; *i.e.*, $(M_1, \ldots, M_n) \leq (N_1, \ldots, N_n)$ iff $M_i \leq N_i$ for all i. An *order view* $\Gamma = (\mathbf{V}_\Gamma, \mu_\Gamma)$ of $\mathbf{D}$ is a view with the property that $\mathbf{V}$ is an order schema and γ' is an order-preserving function; *i.e.*, for any $M, N \in \mathsf{LDB}(\mathbf{D})$, $M \leq N$ implies that $\gamma'(M) \leq \gamma_n{}'$. The subdirect decomposition X of the set-based schema $\mathbf{D}$ into order views is an *order decomposition* if $\Delta\langle X\rangle$ is a section (isomorphism into); *i.e.*, for any $M, N \in \mathsf{LDB}(\mathbf{D})$, $\Delta\langle X\rangle'(M) \leq \Delta\langle X\rangle'(N)$ iff $M \leq N$.

3.3 Full and pairwise reduction Now assume that X is an order decomposition of the set-based $\mathbf{D}$. Let $\vec{M} \in \mathsf{LDB}(\mathbf{V}_1) \times \ldots \times \mathsf{LDB}(\mathbf{V}_n)$. We say that $\vec{M}$ has a *full reduction* if there is a largest (under the induced product ordering) $\vec{N} \in \Delta\langle X\rangle'(\mathsf{LDB}(\mathbf{D}))$ with the property that $\vec{N} \leq \vec{M}$. We call $\vec{N}$ the *full reduction* of $\vec{M}$ with respect to X, and use the explicit notation $\mathsf{FullRed}(\vec{M}, X) = (\mathsf{FullRed}(\vec{M}, X, 1), \ldots, \mathsf{FullRed}(\vec{M}, X, n)$ to denote it. It is immediate that this notion generalizes the relational one described in 3.1 above.

Continuing in the same context, let $\{i, j\} \subseteq \{1, \ldots, n\}$, and let $\vec{M} = (M_i, M_j) \in \mathsf{LDB}(\mathbf{V}_i) \times \mathsf{LDB}(\mathbf{V}_j)$. We say that $\vec{M}$ admits a (Γ_i, Γ_j)-*pairwise reduction* if the following two conditions are met.

(i) $\{\Gamma_i, \Gamma_j\}$ is fully commuting.

(ii) There is a largest $\vec{N} = (N_i, N_j) \in \Delta\langle\{\Gamma_i, \Gamma_j\}\rangle'(\mathsf{LDB}(\mathbf{D}))$ such that $\vec{N} \leq \vec{M}$.

We call $N_i \in \mathsf{LDB}(\mathbf{V}_i)$ the (Γ_i, Γ_j)-*pairwise reduction* of M_i by M_j, and denote it by $\mathsf{PairRed}(\vec{M}, \Gamma_i, \Gamma_j)$. In view of Proposition 1.7 and the discussion of 3.1, it follows that in the context of the simple universal relational schema $\mathbf{R}[U_1, \ldots, U_n]$, for any U_i and U_j and $\vec{M} = (M_i, M_j) \in \mathsf{LDB}(\mathbf{R}_{U_i}) \times \mathsf{LDB}(\mathbf{R}_{U_j})$, $\mathsf{PairRed}(\vec{M}, \Pi_{U_i}, \Pi_{U_j})$ always exists, and is just $M_i \ltimes M_j$.

Given $\vec{M} = (M_1, \ldots, M_n) \in \mathsf{LDB}(\mathbf{V}_1) \times \ldots \times \mathsf{LDB}(\mathbf{V}_n)$, $i \in \{1, \ldots, n\}$, and $P \in \mathsf{LDB}(\mathbf{V}_i)$, define the *i-substitution* of P in M to be $\mathsf{Subst}(\vec{M}, i, P) = (M_1, \ldots, M_{i-1}, P, M_{i+1}, \ldots, M_n)$. In other words, we replace M_i with P, and leave the rest of the n-tuple unchanged. Define $\mathsf{Pair}(\vec{M}, i, j)$ to be (M_i, M_j). Finally, given a finite sequence $S = (\Gamma_{i_{11}}, \Gamma_{i_{12}}), \ldots, (\Gamma_{i_{m1}}, \Gamma_{i_{m2}}) \in X \times X$, define, for $0 \leq j \leq m$ and $\vec{M} \in \mathsf{LDB}(\mathbf{V}_1) \times \ldots \times \mathsf{LDB}(\mathbf{V}_n)$,

$$\mathsf{Seqred}\langle S, j\rangle(\vec{M}) = \begin{cases} \vec{M} & \text{if } j = 0; \\ \mathsf{Subst}(\mathsf{Seqred}\langle S, j-1\rangle(\vec{M}), i_{j1}, \\ \qquad \mathsf{PairRed}(\mathsf{Pair}(\mathsf{Seqred}\langle S, j-1\rangle(\vec{M}), i_{j1}, i_{j2}), \Gamma_{i_{j1}}, \Gamma_{i_{j2}})) & \text{if } j > 0. \end{cases}$$

Define $\mathsf{Seqred}\langle S\rangle(M) = \mathsf{Seqred}\langle S, m\rangle(M)$. In words, the above formula just applies the pairwise reductions defined by S to $\vec{M}$ in a sequential fashion. The sequence S is a *pairwise full reducer* for X if $\mathsf{Seqred}\langle S\rangle(M) = \mathsf{FullRed}(M, X)$ for every $\vec{M} \in \mathsf{LDB}(\mathbf{V}_1) \times \ldots \times \mathsf{LDB}(\mathbf{V}_n)$. 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 pairwise full reducer* if such an S exists. We have as our generalization of (S8) the following condition.

(T8) X admits a pairwise full reducer.

3.4 Remarks concerning this generalization In the simple universal relational case, the full reduction of any $\vec{M} \in \mathsf{LDB}(\mathbf{R}_{U_1}) \times \ldots \mathsf{LDB}(\mathbf{R}_{U_n})$ always exists, regardless of whether or not a sequential full reducer exists to compute it. However, in our generalization, we do not stipulate that the full reduction always exist. This is not a weakness of the generalization, but rather a consequence of the fact that constraints can prevent the existence of a full reduction, even in the relational context. Suppose now that, rather than working in the context of the simple universal relational schema $\mathbf{R}[U_1, \ldots, U_n]$, we work with a more complex universal relational schema $\mathbf{R} = (R[U],\{\bowtie [U_1, \ldots, U_n]\} \cup \Phi)$ for which Φ forces the embedding of some tuple-generating dependencies [Fag82] into the view schemata. Then, the process of deleting tuples from one of the $\mathsf{LDB}(\mathbf{R}_{U_i})$'s during the reduction process may yield an illegal state, since a dependency in $\pi_{U_i}{}'(\Phi)$ may fail to be satisfied. Indeed, the full reduction of some n-tuples may not exist. Since such a universal relational schema must be subsumed by any general set-based generalization, we must therefore accomodate the reality that a full reduction in the general case will only exist under certain circumstances. We do this by stipluating that it only need exist in the case that we have a sequential full reducer to compute it.

In any case, we have the following augmentation of the main theorem.

3.5 Extended Theorem 2.15 *Let* $\mathbf{D}$ *be an order schema, and let* X *be an order decomposition of* $\mathbf{D}$*. Then the following may be added as an equivalent condition to Theorem 2.15.*

(T8) *X admits a pairwise full reducer.* $\square$

4. Conclusions and Further Directions

In this paper, we have laid down the foundations for the generalization of the classical notion of relational schema acyclicity to much more general classes of schemata. We have shown that the key characterizations of schema acyclicity are not particular to the relational model, but rather are fundamental properties which define "pairwise" or "two-at-a-time" properties of general schemata. It will be an interesting next step to ascertain what sort of results can be obtained by applying these notions to other types of schemata. Particularly, a few years ago we proposed a generalization of the notion of acyclicity for relational schemata which admit both horizontal and vertical decompositions [Heg88]. Unfortunately, we never published the full details, primarily because they are so complicated. In part, our feeling that such complexity was not inherent to the results has motivated this present work, and it is our conjecture that those same results may be obtained much more easily by applying the present general framework to that context, rather than by the direct methods of that previous paper. However, the details remain to be elaborated.

In [Fag83], Fagin identifies various types of acyclicity, and particularly notes that there is a stronger notion (γ-acyclicity) than the usual one (which is termed α-acyclicity there). γ-acyclicity has some particularly nice properties regarding simplicity of query evaluation. An interesting and potentially useful extension of the work presented here will be to ascertain whether γ-acyclicity has a similar generalization.

References

[Ala89] Alagić, S., *Object-Oriented Database Programming*, Springer-Verlag, 1989.

[BS81] Bancilhon, F. and Spyratos, N., "Update semantics of relational views," *ACM Trans. Database Systems*, **6**(1981), 557–575.

[BFMY83] Beeri, C., Fagin, R., Maier, D., and Yannakakis, M., "On the desirability of acyclic database schemes," *J. Assoc. Comp. Mach.*, **30**(1983), 479–513.

[CGT90] Ceri, S., Gottlob, G., and Tanca, L., *Logic Programming and Databases*, Springer-Verlag, 1990.

[Fag82] Fagin, R., "Horn clauses and database dependencies," *J. Assoc. Comp. Mach.*, **29**(1982), 952–985.

[Fag83] Fagin, R., "Degrees of acyclicity for hypergraphs and relational database schemes," *J. Assoc. Comp. Mach.*, **30**(1983), 514–550.

[FMU82] Fagin, R., Mendelzon, A. O., and Ullman, J. D., "A simplified universal relation assumption and its properties," *ACM Trans. Database Systems*, **7**(1982), 343–360.

[FV86] Fagin, R. and Vardi, M. Y., "The theory of data dependencies – a survey," in: Anshel, M. and Gewirtz, W., eds., *Mathematics of Information Processing*, pp. 19–71, American Mathematical Society, 1986.

[Fle55] Fleischer, I., "A note on subdirect products," *Acta Math. Acad. Sci. Hungar.*, **6**(1955), 463–465.

[GMN84] Gallaire, H., Minker, J., and Nicolas, J., "Logic and databases: a deductive approach," *ACM Comput. Surveys*, **16**(1984), 153–185.

[Gra68] Grätzer, G., *Universal Algebra*, D. Van Nostrand, 1968.

[Gra78] Grätzer, G., *General Lattice Theory*, Academic Press, 1978.

[Heg88] Hegner, S. J., "Decomposition of relational schemata into components defined by both projection and restriction," in: *Proceedings of the Seventh ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pp. 174–183, 1988.

[Heg89] Hegner, S. J., Unique complements and decompositions of database schemata, Technical Report PC 12/ 12.89, Centro di Ricerche in Fisica e Matematica (CERFIM), Locarno, Switzerland, 1989. Submitted for publication.

[Heg90] Hegner, S. J., "Foundations of canonical update support for closed database views," in: Abiteboul, S. and Kanellakis, P. C., eds., *ICDT'90, Third International Conference on Database Theory, Paris, France, December 1990*, pp. 422–436, Springer-Verlag, 1990.

[HS73] Herrlich, H. and Strecker, G. E., *Category Theory*, Allyn and Bacon, 1973.

[Mai83] Maier, D., *The Theory of Relational Databases*, Computer Science Press, 1983.

[PDGV89] Paredaens, J., De Bra, P., Gyssens, M., and Van Gucht, D., *The Structure of the Relational Database Model*, Springer-Verlag, 1989.

[Rei84] Reiter, R., "Towards a logical reconstruction of relational database theory," in: Brodie, M. L., Mylopoulos, J., and Schmidt, J. W., eds., *On Conceptual Modelling*, pp. 191–233, Springer-Verlag, 1984.

[Var82] Vardi, M. Y., "On decompositions of relational databases," in: *Proceedings 23rd Annual Symposium on Foundations of Computer Science*, pp. 176–185, 1982.

[Wen67] Wenzel, G. H., "Note on a subdirect representation of universal algebras," *Acta Math. Acad. Sci. Hungar.*, **18**(1967), 329–333.