Unique Complements and Decompositions of Database Schemata

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, complements

This paper appeared in the *Journal of Computer and System Sciences*, Vol. 48, No. 1, 1994, pp. 9-57.

ABSTRACT

In earlier work, Bancilhon and Spyratos introduced the concept of a complement to a database schema, and showed how this notion could be used in theories of decomposition and update semantics. However, they also showed that, except in trivial cases, even minimal complements are never unique, so that many desirable results, such as canonical decompositions, cannot be realized. Their work dealt with database schemata which are sets and database mappings which are functions, without further structure. In this work, we show that by adding a modest amount of additional structure, many important uniqueness results may be obtained. Specifically, we work with database schemata whose legal states form partially ordered sets (posets) with least elements, and with database mappings which are isotonic and which preserve this least element. This is a natural algebraic structure which is inherent in many important examples, including relational schemata constrained by data dependencies, with views constructed by composition of projection, restriction, and selection. Other examples include deductive database schemata in which views are defined by rules, and general first-order logic databases.

Within this context of posets, we show that direct (*i.e.*, independent) complements must be unique, and that in fact the directly complementable views have the structure, in a very natural sense, of a Boolean algebra. Decompositions of the schema then become identifiable with finite subalgebras of this Boolean algebra. To demonstrate the utility of our approach, we examine in some detail its applicability to the relational model. Particularly, we establish that under the condition that the schema is constrained by universal Horn sentences, there is a unique ultimate decomposition into a finite set of type restrictions. The latter are a special class of views which includes classical projections which occur in direct decompositions. In particular, classical join-based decomposition is completely recovered within a framework which explicitly axiomatizes independence via null values.

0. Introduction

0.1 Motivation and Overview

The notion of decomposition of a database schema has long been identified as an important one. With the development of the relational approach over the past two decades, various decomposition theories for this model have been developed. In early work on the relational model, Codd [13] showed that certain types of anomalies in data representation could be avoided by *normalization*, a particular form of decomposition. Subsequently, a more general theory of relational schema decomposition arose from the theories of joins [1] and acyclicity [18], as did more elaborate theories of normalization [16, 17]. These approaches all share the common feature that they address the decomposition of a very specific class of schemata, namely relational schemata constrained by functional and join dependencies.

In this paper, we provide a complementary approach to decomposition of database schemata which is not restricted to a specific class of schemata, but which rather makes the fewest assumptions necessary to support the results. Such a general theory is important for several reasons. First of all, while the traditional relational model has remained an important one, there has recently been much interest in more sophisticated data models, such as object-oriented models [3], deductive models [42, 11], and incomplete-information models [20, 45]. The specific relational decomposition theories cited above do not extend in an obvious way to any of these other models. As these alternate data models mature, specific decomposition theories will likely arise. However, such theories should not start from scratch, ignoring those results already achieved in the traditional relational case. Rather, they should be based upon whatever general principles the relational theory can provide. To do so, a better understanding of how the specific relational decomposition theories fit into a general framework is necessary. Second, within the relational model itself there are other concepts of decomposition, such as horizontal decomposition [46, 14], which deserve to be better understood. And finally, the notion of decomposition has been shown to be intimately related to the support of view updates via the *constant complement strategy* [5], and so with a better understanding of decomposition will we acquire a better understanding of the difficult problem of view update.

We address primarily *directly* complemented views; that is, views which are the constituents in decompositions of schemata into *independent* components. Our approach is motivated by the decomposition theory of Bancilhon and Spyratos [4, 5], in which database schemata are just sets and views are surjective functions. The limitation of their framework is that it is *too* general; it is impossible to obtain any kind of useful uniqueness theorems because too many mappings are admitted as views. In particular, complements are unique only in trivial cases. The key is to add just the right amount of additional structure. Our proposal is that this amount is precisely to require that each database schema have the structure of a $\perp$-poset; that is, a partially ordered set with a least element. Database morphisms are required to be order and least-element preserving. Many examples of interest possess this additional structure, including relational schemata constrained by *data dependencies* in the sense of [19] with project-restrict-join database morphisms, as well as logic databases ordered under theory containment and deductive databases. With this modest amount of additional structure, we show that direct complements are unique when they exist, and that any two decompositions have a coarsest unique refinement. When the number of directly complemented views is finite, we furthermore

have unique ultimate decompositions.

The approach presented in this paper is distantly related to our earlier work on relational decomposition [26, 27]. However, the current work is simpler and at the same time applicable to a wide variety of schema types, because we formulate prove our main results while working solely with the underlying order-theoretic structure. The specific applications, such as to the relational theory, are then built on top of this framework. Our previous work, on the other hand, dealt directly with the relational model.

The only other work of which we are aware which uses order-theoretic concepts to construct complements is that of Keller and Ullman [37], who assume that the states of the database schema form not only a $\perp$-poset, but in fact a finite Boolean algebra. While such an assumption easily leads to uniqueness results, it is also far too strong to be of any practical value. Very few database applications admit a model in which the legal databases form a Boolean algebra. In this paper, we are successful in recapturing the main results of [37], within a far less constraining and more realistic framework.

As we have already noted, this paper deals exclusively with direct complements, in which the views are independent. A comparable theory for subdirect complements, in which the component views are dependent upon one another, requires a somewhat different approach. This is discussed further in Part 3.

The paper is divided into two principal parts. Part 1 contains the development of the general theory of database decomposition in a general setting, assuming only the simple order-based algebraic structure mentioned above. The cornerstone result on decomposition structure states that direct complements are unique when they exist, and that the set of all complemented views of a schema forms a Boolean algebra, with view complementation corresponding to complementation in the algebra. Furthermore, the direct decompositions are in bijective correspondence with the atomic Boolean subalgebras of this algebra. It is also shown that the views which participate in a decomposition have the very special structure (up to isomorphism) of an *ideal view*. Roughly speaking, this means that the view must a subschema (as well as a quotient schema) of the main schema which is being decomposed.

Part 2 contains selected applications of the theory of Part 1 to the relational model. Our intent is not to provide a complete theory of relational decomposition, as that would constitute several papers in itself. Rather, the direction of this section is guided by an interest in providing a basic understanding how the traditional theory of join-based decomposition fits into the larger picture. In harmony with our earlier work [26, 27, 29], we work with a Boolean algebra of domain types, rather than a set of disjoint domains. Such a framework is not merely a superficial extension of the traditional one; rather, it is shown to be essential in the representation of null values, which are a necessary part of any join-based decomposition into independent components.

Section 2.1 provides the essential foundations for our approach. The key result is that under the assumption that the schema to be decomposed is governed by universal Horn constraints (which include all *total dependencies* in the sense of [19]), each view in any direct decomposition is *tuple based*. This means that the view mapping functions need only look at one tuple at a time; the computation of the view does not depend upon combinations of tuples being present or absent.

Section 2.2 examines decomposition into a particularly simple kind of tuple-based view, the *type restrictions*, in which the the only relevant property of a tuple, for view computation

purposes, is the type domain of each of its entries. The key result here is that if we impose the additional constraint corresponding to the property of a dependency being *typed* [19], then decompositions governed by universal Horn sentences must be type restrictions. We further show that the set of all such type restrictions is finite, thus establishing that any schema satisfying the above conditions has a unique ultimate decomposition into a finite set of type restrictions. To illustrate the power of this representation, we show in particular how the classical notion of a decomposition into projections based upon a join dependency may be recaptured in this framework. This representation makes very clear how to formally represent the use of nulls to achieve true independent decomposition, a topic which has been treated informally for many years, but has only recently seen formalization, even within the specific join- and functional-dependency governed framework [12].

An important concept in the use of nulls in decomposition is tuple subsumption. For example, if ν is a null and c is an ordinary domain value, we may say that the tuple (a, b, ν) is subsumed by the tuple (a, b, c). In Section 2.3, we show how to explicitly incorporate this sort of subsumption into the order-based relational model. The use of such subsumption provides models of join-based decomposition which are perhaps more aesthetic than those of Section 2.2, since subsumed tuples need not be represented. We show, however, that this is merely a surface difference, as we prove a representation theorem which provides, for any subsumption-based model, an equivalent (both logically and order theoretically) type-restriction based model. This is a key result, because while models with tuple subsumption may be more aesthetic, type-restriction based models are much easier to manipulate mathematically.

Section 2.4 provides a very brief discussion of horizontal decomposition within our framework.

0.2 Prerequisites and Notation

While the mathematical results developed herein may be understood on a formal level with little or no knowledge of database theory, the motivation for the direction of the paper, as well as an understanding of the examples, requires a fundamental knowledge of the terminology, notation, and principal results of the theory of relational databases, such as may be found in [2], [39], [44], and [47]. For Part 2 of this paper, we also assume some basic knowledge of first-order logic, as may be found in [15], [22], and [43]. However, except for a few proofs which may be skipped without loss of continuity, only a rather basic knowledge of syntax and semantics is essential, and an understanding of the survey article [21] should prove sufficient.

The notions of partial order, lattice, and Boolean algebra are used extensively, although no knowledge of any but the most fundamental definitions and results is assumed without explicit reference. The appropriate definitions and background may be found in [25] and [8].

1. The Theory of Order-Based Decomposition

In this part, we present a general theory of schema decomposition within the context of $\perp$-posets. While we give some motivating examples from the standard relational approach, the general results are not rooted in the relational model in any way.

1.1 Schemata and Views

1.1.1 Motivating example – the property of isotonicity Let $\mathbf{F}$ be the relational schema with only one domain A and exactly two unary relational symbols $R[A]$ and $S[A]$. There are no constraints, other than that the two relations share the same domain A. We let $\Gamma_R = (R[A], \pi_R)$ and $\Gamma_S = (S[A], \pi_S)$ denote the views which preserve identically the named relation, and discard the other. Under any reasonable definition, $\{\Gamma_R, \Gamma_S\}$ forms a decomposition of $\mathbf{F}$ into independent components. Indeed, the state of $\mathbf{F}$ can be trivially recovered from the combined states of Γ_R and Γ_S, and these two views can be updated completely independently, with their resulting states always defining a unique state of $\mathbf{F}$. Furthermore, barring any decomposition based upon additional structure of A (such as horizontal decomposition [46]), it seems clear that this is the only "reasonable" nontrivial decomposition.

It is quite easy, however, to produce another decomposition totally within the relational framework. For any particular database M of the schema $\mathbf{F}$, denote the corresponding relational instances of R and S by R^M and S^M, respectively. Define the view $\Gamma_T = (T[A], \gamma_T)$ to have a single unary relation symbol $T[A]$, with the view mapping $\gamma_T : (R^M, S^M) \mapsto (R^M \cup S^M) \setminus (R^M \cap S^M)$. In other words, the state of $T[A]$ is the symmetric difference of the states of $R[A]$ and $S[A]$. It is easy to see that the state of $\mathbf{F}$ may be uniquely recovered by knowing the states of both Γ_R and Γ_T; indeed, we can recover the state of $S[A]$ by computing the symmetric difference of the relation instances of $R[A]$ and $T[A]$. Furthermore, Γ_R and Γ_T may be independently updated, with the instance of $S[A]$ tracking as the symmetric difference.

In terms of preserving the structure of the schema $\mathbf{F}$ as well as the data contained in the particular instance, the decomposition $\{\Gamma_R, \Gamma_S\}$ seems more natural than $\{\Gamma_R, \Gamma_T\}$. Each of the views Γ_R and Γ_S is a simple projection, while Γ_T makes use of a more complex operator, symmetric difference. The key abstraction here is that projection is a *isotonic* operator; as the state of the base schema $\mathbf{F}$ becomes smaller (under the ordering defined by relation-by-relation inclusion), so does the state of any projection. However, symmetric difference is not isotonic; if we insert tuples into the state of the relation $R[A]$ while keeping $S[A]$ constant, the state of $T[A]$ must become smaller. Formally, Γ_R and Γ_S are *isotonic* views, while Γ_T is not.

Many of the naturally occurring views, at least in the relational theory, are isotonic (*e.g.*, projection, restriction, join), and a reasonable decomposition theory could well consider only such views. Indeed, the main decomposition theories of the relational model have used only projection as a view-defining operation. Thus, while we do not claim that isotonic views are the only important ones, they are sufficiently important that a theory restricted to them (as we shall present in this paper) is of sufficient interest provided that it can provide important results not obtainable in more general contexts.

1.1.2 Motivating example – the property of order isomorphism In this example, we alter the previous example slightly by formally adding the relational symbol T to the base schema to be decomposed. More precisely, let $\mathbf{G}$ be the relational schema with domain A and the three relational symbols $R[A]$, $S[A]$, and $T[A]$. The only constraint states that in any instance $M = (R^M, S^M, T^M)$, any domain element $a \in A$ must occur in none of $\{R^M, S^M, T^M\}$, or in exactly two of them. In other words, any relation instance is the symmetric difference of the other two. Each of the three views $\Gamma_R = (R[A], \pi_R)$, $\Gamma_S = (S[A], \pi_S)$, and $\Gamma_T = (T[A], \pi_T)$ retain the named relation and discard the others. Now all three of the views $\{\Gamma_R, \Gamma_S, \Gamma_T\}$ are

isotonic, and any two of them form a decomposition into independent components; the third can always be recovered from the other two by computing the symmetric difference. Thus, there are three structurally identical decompositions.

The problem here is not with the views, which are all isotonic, but rather with the schema $\mathbf{G}$ itself. The symmetric difference constraint creates a situation in which a deletion to one relation must result in an insertion to another. We rule out such decompositions by requiring that the decomposition map be not only a bijection, but an order isomorphism as well. To illustrate via a specific example, suppose that M is the database state of $\mathbf{G}$ with $R^M = \{a, b\}$, $S^M = \{a\}$, $T^M = \{b\}$. Then a decomposition of $\mathbf{G}$ into (Γ_R, Γ_S) would send this state $(\{a, b\}, \{a\}, \{b\})$ to $(\{a, b\}, \{a\})$. Now if we delete a from R^M, the new state of the decomposition becomes $(\{b\}, \{a\})$ (a smaller state under the natural relation-by-relation ordering). We must reflect this by the base schema state $(\{b\}, \{a\}, \{a, b\})$, which is not smaller than the original state $(\{a, b\}, \{a\}, \{b\})$. Therefore the decomposition map is not an order isomorphism.

We now proceed to the formal definitions.

1.1.3 Schemata and morphisms A *$\perp$-poset database schema* $\mathbf{D}$ is a triple $(\mathsf{LDB}(\mathbf{D}), \leq_{\mathbf{D}}, \perp_{\mathbf{D}})$, in which $\mathsf{LDB}(\mathbf{D})$ is a set, called the *legal databases* of $\mathbf{D}$, $\leq_{\mathbf{D}}$ is a partial order relation[1] on $\mathsf{LDB}(\mathbf{D})$ with a least element, and $\perp_{\mathbf{D}}$ explicitly identifies that least element. Usually, we shall abbreviate $\leq_{\mathbf{D}}$ to just $\leq$, and $\perp_{\mathbf{D}}$ to $\perp$, since context will make clear which subscripts are appropriate. Additionally, since we shall consider only $\perp$-poset database schema in Part 1 of this paper, the term *database schema* shall, by default, mean $\perp$-poset database schema (possibly with additional structure) throughout this part.

Let $Y = \{\mathbf{D}_i \mid i \in I\}$ be an arbitrary set of database schemata, indexed by some set I. The *product schema* of Y, denoted $\prod Y$ or $\prod_{i \in I} \mathbf{D}_i$, has as its underlying set the cartesian product $\prod_{i \in I} \mathsf{LDB}(\mathbf{D}_i)$, with ordering $(x_i)_{i \in I} \leq (y_i)_{i \in I}$ if and only if $x_i \leq y_i$ for each $i \in I$. The least element is $(\perp_{\mathbf{D}_i})_{i \in I}$. If Y is empty, the product schema $\prod Y$ is taken to be the *zero schema* $\mathbf{0}$, whose underlying set is the singleton $\{\perp\}$.

Now let $\mathbf{D}_1 = (\mathsf{LDB}(\mathbf{D_1}), \leq, \perp)$ and $\mathbf{D}_2 = (\mathsf{LDB}(\mathbf{D_2}), \leq, \perp)$ be database schema. A morphism f from $\mathbf{D}_1$ to $\mathbf{D}_2$ is a function $f' : \mathsf{LDB}(\mathbf{D}_1) \to \mathsf{LDB}(\mathbf{D}_2)$ which is preserves the least element and which is isotonic. More formally, we require the following two properties.

($\perp$p-i) $f'(\perp) = \perp$.

($\perp$p-ii) $(\forall x, y \in \mathsf{LDB}(\mathbf{D}_1))(x \leq y \Rightarrow f'(x) \leq f'(y))$

We write $f : \mathbf{D}_1 \to \mathbf{D}_2$ to denote that f is such a morphism, and because it is sometimes essential to distinguish the morphism from its underlying function, the prime superscript will always be used to identify the latter. It is immediate the the composition of morphisms is a morphism. We define *section*, *retraction*, and *isomorphism* in the categorical sense [33]. More precisely, $f : \mathbf{D}_1 \to \mathbf{D}_2$ is a *section* if it has a *left inverse*; *i.e.*, there is a morphism $g : \mathbf{D}_2 \to \mathbf{D}_1$ such that $g \circ f$ is the identity on $\mathbf{D}_1$. The morphism f is a *retraction* if it has a *right inverse*; *i.e.*, if there is such a g such that $f \circ g$ is the identity on $\mathbf{D}_2$. Finally, f is an *isomorphism* if it is both a section and a retraction.

The following characterizations are easily verified.

[1] We always assume partial orders to be reflexive; *i.e.*, $x \leq x$ for any x.

1.1.4 Proposition *Let* $\mathbf{D}_1$ *and* $\mathbf{D}_2$ *be database schemata, and let* $f : \mathbf{D}_1 \to \mathbf{D}_2$ *be a morphism.*

(a) *f is a section if and only if it is injective, and, for each pair $x, y \in \mathsf{LDB}(\mathbf{D}_1)$, $x \leq y$ if and only if $f'(x) \leq f'(y)$.*

(b) *f is a retraction if and only if it is surjective and for each pair $x, y \in \mathsf{LDB}(\mathbf{D}_2)$ with $x \leq y$, there are $z \in (f^{-1})'(\{x\})$ and $w \in (f^{-1})'(\{y\})$ such that $z \leq w$.*

(c) *f is an isomorphism if and only if it is a surjective section.*

□

We now interpret our definitions of schema and view in terms of three common database frameworks.

1.1.5 Example – standard relational interpretation In the relational model, the natural ordering is relation-by-relation inclusion. Typically, relational schemata are assumed to be constrained by so-called *dependencies*, which are sentences of the following form.

$$(\forall x_1, x_2, \ldots, x_r)((A_1 \wedge A_2 \wedge \ldots \wedge A_m) \Rightarrow (\exists y_1, y_2, \ldots, y_r)(B_1 \wedge B_2 \wedge \ldots \wedge B_n))$$

Here the A_i's are relational atoms, and the B_i's are either relational atoms or else statements of equality between terms. It is argued in [19] that most reasonable relational constraints are of this form. If we work with schemata constrained by such constraints, then we immediately have property ($\perp$p-i), since the empty model (all relations empty) is a model of any such dependency, and the empty model is then $\perp$.

As for the isotonicity condition ($\perp$p-ii), we have already observed in 1.1 that the most common database mappings, such as projection, restriction, and join are isotonic. Only those which involve negation, such as difference, are not.

In Section 2.3, we shall formally extend the relational model to include nulls, and define another ordering relation based upon tuple subsumption, much in the same spirit as the work of Zaniolo [50]. In that context, we establish that the conditions of 1.1.3 are met as well.

1.1.6 Example – deductive database interpretation Datalog is a recursive query language for the relational model [11, 10]. The notion of a relational schema is the same as in the traditional case, but a database mapping (or query) from schema $\mathbf{D}$ to schema $\mathbf{V}$ in Datalog is a set of universal Horn clauses of the form

$$\begin{array}{rcl} C_1 & \leftarrow & C_{11}, .., C_{1n_1}; \\ C_2 & \leftarrow & C_{21}, .., C_{2n_2}; \\ .. & & .. \\ C_k & \leftarrow & C_{k1}, .., C_{1n_k}. \end{array}$$

The C_i's use relation symbols from $\mathbf{V}$, while the C_{ij}'s use relation symbols from both $\mathbf{D}$ and $\mathbf{V}$. The key observation is that because all literals are positive, a larger database for $\mathbf{D}$ will result in a larger image computed for $\mathbf{V}$. Isotonicity is guaranteed; indeed, a cornerstone of this entire framework is that the query may be computed as the least fixpoint of the above operators. Preservation of the least model is immediate as well, so our decomposition framework applies to the deductive database setting.

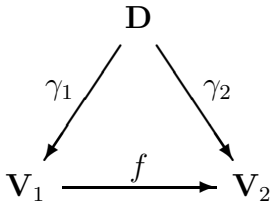
1.1.7 Example – incomplete information database interpretation In an incomplete information schema, a database is not a single structure satisfying the constraints, but rather a set of such structures. Usually, such a database is specified by a set of first order sentences, and the set of structures is just the set of models of those sentences [21, 45], but there are certainly other possibilities. In any case, let $\mathbf{D}$ be a database schema, and let $\mathsf{CLDB}(\mathbf{D})$ denote the set of legal complete-information databases of $\mathbf{D}$, in the usual sense. Let $\mathsf{ILDB}(\mathbf{D})$ denote the set of incomplete-information databases, which is exactly the set of all subsets of $\mathsf{CLDB}(\mathbf{D})$. We order $\mathsf{ILDB}(\mathbf{D})$ under reverse inclusion; that is, $S_1 \leq S_2$ if $S_2 \subseteq S_1$. Larger elements in this ordering contain more information, hence fewer possible worlds. The least element is the set of all subsets of $\mathsf{CLDB}(\mathbf{D})$.

We take morphisms in this context to be induced by the complete-information case. That is, $f : \mathbf{D}_1 \to \mathbf{D}_2$ is a morphism if there is an underlying function $g : \mathsf{CLDB}(\mathbf{D}_1) \to \mathsf{CLDB}(\mathbf{D}_2)$ such that $f'(S) = \{g(s) \mid s \in S\}$. In general, such a mapping will not be least-element preserving. However, in the special case that g is surjective, it clearly will be. As will be seen as our presentation unfolds, we work exclusively with morphisms whose underlying functions are surjective. Hence, in the cases that we consider, this definition yields a valid $\perp$-poset morphism.

Thus, the incomplete information setting fits into our general decomposition framework as well.

1.1.8 Views and view morphisms Given a database schema $\mathbf{D}$, a *view* of $\mathbf{D}$ is a pair $\Gamma = (\mathbf{V}, \gamma)$ in which $\mathbf{V}$ is a database schema and $\gamma : \mathbf{D} \to \mathbf{V}$ is a morphism whose underlying function f' is surjective. We call $\mathbf{D}$ the *base schema*, $\mathbf{V}$ the *view schema*, and γ the *view mapping*. The collection of all views of $\mathbf{D}$ is denoted $\mathsf{View}(\mathbf{D})$.

Given views $\Gamma_1 = (\mathbf{V}_1, \gamma_1)$ and $\Gamma_2 = (\mathbf{V}_2, \gamma_2)$ of $\mathbf{D}$, a *view morphism* $f : \Gamma_1 \to \Gamma_2$ is a database morphism $f : \mathbf{V}_1 \to \mathbf{V}_2$ such that the following diagram commutes.



f is an isomorphism of views when it has a left and right inverse. It is immediate that view isomorphism induces an equivalence relation on $\mathsf{View}(\mathbf{D})$. We let $[\mathsf{View}(\mathbf{D})]$ denote the partition on $\mathsf{View}(\mathbf{D})$ induced by this equivalence. That is, two views are in the same block of $[\mathsf{View}(\mathbf{D})]$ if and only if they are isomorphic.

We also have the following important observation.

1.1.9 Lemma – uniqueness of view morphisms *Given two views $\Gamma_1 = (\mathbf{V}_1, \gamma_1)$ and $\Gamma_2 = (\mathbf{V}_2, \gamma_2)$ of the database schema $\mathbf{D}$, there is at most one morphism $f : \Gamma_1 \to \Gamma_2$, and f' is necessarily surjective.*

PROOF: These are actually special cases of more general facts from category theory ([33, 5.11, 6.8]), but we give a direct proof. Let $f, g : \Gamma_1 \to \Gamma_2$ each be view morphisms, and let

$x \in \mathsf{LDB}(\mathbf{V}_1)$. Since ${\gamma_1}'$ is surjective, there is a $y \in \mathsf{LDB}(\mathbf{D})$ with ${\gamma_1}'(y) = x$. But then $f'(x) = g'(x) = {\gamma_2}'(y)$, whence $f = g$. Finally, since ${\gamma_2}'$ is surjective, it is immediate that f' must be as well. □

1.1.10 Special schemata and views The morphism guaranteed by the above lemma is of sufficient importance to warrant a special notation. When it exists, we denote the unique f which makes the above diagram commute by $\lambda(\Gamma_1, \Gamma_2)$. This furthermore allows us to regard Γ_2 as a view of $\mathbf{V}_1$. We call this new view the *relativization* of Γ_2 to Γ_1, and denote it by $\Lambda(\Gamma_1, \Gamma_2) = (\mathbf{V}_2, \lambda(\Gamma_1, \Gamma_2))$. In [30], more specific results regarding the structure of relative views is provided.

There are two special views of a schema $\mathbf{D}$ which we will need often. The *identity view*, denoted $\Gamma_\top(\mathbf{D}) = (\mathbf{D}, \mathbf{1_D})$, preserves identically the base schema $\mathbf{D}$; the view mapping ${\mathbf{1_D}}'$ is the identity on $\mathsf{LDB}(\mathbf{D})$. The *zero view*, denoted $\Gamma_\bot(\mathbf{D}) = (\mathbf{0}, \mathbf{0_D})$, has as its underlying schema the zero schema $\mathbf{0}$. The view mapping $\mathbf{0'_D}$ sends all elements of $\mathsf{LDB}(\mathbf{D})$ to the state $\bot$.

1.2 The Decomposition Morphism and Types of Decompositions

1.2.1 Notational convention Unless otherwise noted, throughout Sections 1.2 and 1.3, we let $\mathbf{D}$ denote a $\bot$-poset database schema, and $X = \{\Gamma_i \mid i \in I\}$ an arbitrary finite set of views of $\mathbf{D}$, with $\Gamma_i = (\mathbf{V}_i, \gamma_i)$.

Our goal is to identify the properties which X must have in order that it be a decomposition of $\mathbf{D}$. To begin, we must formally identify the meaning of decomposition.

1.2.2 The decomposition morphism of a set of views The *decomposition morphism* $\Delta\langle X\rangle : \mathbf{D} \to \prod X$ has as underlying function $\Delta\langle X\rangle' : \mathsf{LDB}(\mathbf{D}) \to \prod_{i\in I} \mathsf{LDB}(\mathbf{V}_i)$, given on elements by $s \mapsto ({\gamma_i}'(s))_{i\in I}$. (It is trivial to verify that $\Delta\langle X\rangle$ is indeed a morphism.) When X is the empty set, we take $\Delta\langle X\rangle$ to be the underlying mapping $\mathbf{0_D} : \mathbf{D} \to \mathbf{0}$ of the zero view $\Gamma_\bot(\mathbf{D})$.

In any reasonable decomposition of a schema $\mathbf{D}$, we demand that the state of $\mathbf{D}$ be recoverable from the collective states of the components of the decomposition. There are, however, two important variants on this theme. In a *subdirect decomposition*, the individual components of the decomposition may be interrelated. In a *direct decomposition*, on the other hand, the individual components of the decomposition must be independent of one another. In the terminology of [6], subdirect decomposition mandates the *representation principle*, while direct decomposition mandates both the representation principle and the *separation principle*.

1.2.3 The types of decomposition
(a) X is a *subdirect decomposition* of $\mathbf{D}$ if $\Delta\langle X\rangle$ is a $\bot$-poset section.
(b) X is a *direct decomposition* of $\mathbf{D}$ if $\Delta\langle X\rangle$ is a $\bot$-poset isomorphism.
(c) X is *independent* if $\Delta\langle X\rangle$ is a $\bot$-poset retraction.

In the case that X is a decomposition (either kind), the left inverse of $\Delta\langle X\rangle$ is called the *reconstruction map*, because it provides the formula for reconstructing the state of the original schema from those of the component views in the decomposition.

Although independence is not a type of decomposition, we have included the definition because of its importance in the formulation of decomposition. Intuitively, independence of a set of views means that there are no constraints relating their states; any of the views may take on any of its states regardless of the states of the others. Note that a direct decomposition is precisely a subdirect decomposition whose elements are independent.

We have borrowed the adjectives *direct* and *subdirect* from the field of universal algebra. The interested reader is invited to compare our definitions with those of *direct product* and *subdirect product* as given in [24].

To crystallize the significance of these forms of decomposition, we illustrate them in terms of a familiar example.

1.2.4 Example Let $\mathbf{E}$ be the relational schema $R[ABCD]$, constrained by the join dependency $\bowtie [AB, BC, CD]$. For each member Z of $\{A, B, C, AB, BC, CD, ABC, BCD, ABCD\}$, let Γ_Z denote the view $(R[Z], \pi_Z)$. These views compute the obvious projections on $\mathbf{E}$. For example, $\Gamma_{AB} = (R[AB], \pi_{AB})$ computes the AB projection.

The family $S_1 = \{\Gamma_A, \Gamma_B, \Gamma_C, \Gamma_D\}$ is not either kind of decomposition, since we cannot recover $R[ABCD]$ from its unary projections. On the other hand, by the classical theory of join dependencies [1], the decomposition of $\mathbf{E}$ into $S_2 = \{\Gamma_{ABC}, \Gamma_{BCD}\}$ satisfies the representation principle, since the join dependency $\bowtie [ABC, BCD]$ is entailed by $\bowtie [AB, BC, CD]$, and so we can reconstruct $R[ABCD]$ from these projections by computing their join. Hence S_2 is a subdirect decomposition of $\mathbf{E}$. However, it is not a direct decomposition, because the views are interrelated; they have the "common component" Γ_{BC} embedded within them. In any decomposition, the view states must agree on these common components.

Consider now the family $S_3 = \{\Gamma_{AB}, \Gamma_{BC}, \Gamma_{CD}\}$. This is also a subdirect decomposition, since the join dependency $\bowtie [AB, BC, CD]$ allows us to reconstruct $R[ABCD]$ from these three projections. However, this is not a direct decomposition either. Indeed, Γ_{AB} and Γ_{BC} have the common component Γ_B, while Γ_{BC} and Γ_{CD} have the common component Γ_C. It has been well known for many years in relational decomposition theory that the judicious use of null values can make S_3 independent. However, it was not until comparatively recently that this was adequately formalized in the literature, even for the special case of join and functional dependencies as the only constraints [12]. We raise this point here because our general approach requires that such independence be completely formalized. It cannot differentiate between the type of inter-view dependence in S_2 and that in S_3, because that distinction lies in subtleties of the relational model, and not in the more global mathematical properties of decomposition. In Sections 2.2 and 2.3, we show how to formalize this notion of independence in the relational model without any recourse to specific constraints.

It is interesting to note that no subset of $\{\Gamma_z \mid z \in Z\}$ containing more than one element is independent, since if one of the projections is empty, then they must all be empty. To achieve independence in the relational model, we must either allow nulls, or else take views of a multirelational schema. (For example, if we were to add a second relation $S[DEF]$ to $\mathbf{E}$, with no interrelational constraints, then clearly any projection on $R[ABCD]$ is independent of any projection of $S[DEF]$. Also, the views Γ_R and Γ_S of 1.1.1 are clearly independent.)

1.2.5 Complements A complement of a view Γ_1 is nothing more than a second view Γ_2 with the property that $\{\Gamma_1, \Gamma_2\}$ is a decomposition. More precisely, let Γ_1 and Γ_2 be views of the schema $\mathbf{D}$. Γ_2 is a *subdirect complement* of Γ_1 if $\{\Gamma_1, \Gamma_2\}$ forms a subdirect decomposition of $\mathbf{D}$. Similarly, Γ_2 is a *direct complement* of Γ_1 if $\{\Gamma_1, \Gamma_2\}$ forms a direct decomposition of $\mathbf{D}$. We also say in these cases that $\{\Gamma_1, \Gamma_2\}$ forms a subdirect or direct *complementary pair*, as the case may be.

In [4, 5], the term *complement* is used to define what we call a subdirect complement. Bancilhon and Spyratos do not use a special term for what we call a direct complement. We have not followed their terminology because, in the algebraic framework in which we will work, the usual meaning of "complement" will correspond to what we have defined to be a direct complement, and not subdirect complement.

1.3 Algebraic Characterization of Direct Decomposition

In this section, we develop the general properties of direct decompositions, and prove that direct complements are unique. The key observation is that, within the $\perp$-poset context, direct complements have a very special structure. The overall idea is as follows. Let $\{\Gamma_1, \Gamma_2\}$ be a direct complementary pair of $\mathbf{D}$, with $\Gamma_1 = (\mathbf{V}_1, \gamma_1)$ and $\Gamma_2 = (\mathbf{V}_2, \gamma_2)$. Since the decomposition mapping $\Delta\langle\{\Gamma_1, \Gamma_2\}\rangle$ is an isomorphism, we may form its inverse $(\Delta\langle\{\Gamma_1, \Gamma_2\}\rangle)^{-1} : \mathbf{V}_1 \times \mathbf{V}_2 \to \mathbf{D}$. Now if we restrict $(\Delta\langle\{\Gamma_1, \Gamma_2\}\rangle)^{-1}$ to $\mathbf{V}_1 \times \mathbf{0}$, we get a natural embedding of $\mathbf{V}_1$ in $\mathbf{D}$. This embedding is the least left inverse of γ_1 in the natural ordering of morphisms ($f_1 \leq f_2$ if and only if $f_1(x) \leq f_2(x)$ for all x in the domain). Thus, $\mathbf{V}_1$ may be regarded as a natural subschema of $\mathbf{D}$. Similarly, $\mathbf{V}_2$ may be regarded as such a subschema.

Now let $\Gamma_3 = (\mathbf{V}_3, \gamma_3)$ be another direct complement of Γ_1. Since $\mathbf{V}_2$ is a subschema of $\mathbf{D}$, we may use $\{\Gamma_1, \Gamma_3\}$ to decompose $\mathbf{V}_2$ as well. But since $\{\Gamma_1, \Gamma_2\}$ is also a direct complementary pair, we have that Γ_1 and Γ_2 are independent, and so the decomposition of $\mathbf{V}_2$ using $\{\Gamma_1, \Gamma_3\}$ must yield the decomposition $\mathbf{0} \times \mathbf{V}_3$. In other words, the natural embedding of $\mathsf{LDB}(\mathbf{V}_3)$ in $\mathsf{LDB}(\mathbf{D})$ must be a subset of the natural embedding of $\mathsf{LDB}(\mathbf{V}_2)$ in $\mathsf{LDB}(\mathbf{D})$. Similarly, the natural embedding of $\mathsf{LDB}(\mathbf{V}_2)$ in $\mathsf{LDB}(\mathbf{D})$ must be a subset of the natural embedding of $\mathsf{LDB}(\mathbf{V}_3)$ in $\mathsf{LDB}(\mathbf{D})$, whence they must be equal. Translating back to the original (non-ideal) structure of the views Γ_2 and Γ_3, we have that these views must be isomorphic.

Of course, we have omitted many details in the preceding sketch, and the goal of this section is to fill them in. We start by characterizing the algebraic structure which the states of a directly complemented view will have. They form what we term a complete ideal, which is a property closely related to that of an ideal in a lattice [25]. This permits us to restrict our attention to decompositions into views with this very special and powerful embedding property, which we call ideal views.

1.3.1 Ideals and ideal views A subset $J \subseteq \mathsf{LDB}(\mathbf{D})$ is called a *complete ideal* of $\mathbf{D}$ if the following two conditions are satisfied.

(ci-i) $x \in J$ and $y \leq x$ implies $y \in J$.

(ci-ii) $x \in \mathsf{LDB}(\mathbf{D})$ implies $\sup(\{y \in J \mid y \leq x\})$ exists in $\mathsf{LDB}(\mathbf{D})$ and is itself in J.

Note that $\sup(\emptyset) = \perp$, so that $\perp$ is a member of every complete ideal. The set of all complete ideals of $\mathbf{D}$ is denoted $\mathsf{Ideal}(\mathbf{D})$. The complete ideal J is called *principal* if there is an $a \in J$

such that $J = \{x \in \mathsf{LDB}(\mathbf{D}) \mid x \leq a\}$. We denote such a principal ideal by $\langle a \rangle$.

An *ideal subschema* of $\mathbf{D}$ is a schema $\mathbf{J}$ in which $\mathsf{LDB}(\mathbf{J})$ is a complete ideal of $\mathbf{D}$. The morphism $\epsilon\langle \mathbf{D}, \mathbf{J} \rangle : \mathbf{D} \to \mathbf{J}$ is given on elements by $x \mapsto \sup(\{y \in \mathsf{LDB}(\mathbf{J}) \mid y \leq x\})$. Note in particular that $\epsilon\langle \mathbf{D}, \mathbf{J} \rangle'$ is quasi-contracting ($\epsilon\langle \mathbf{D}, \mathbf{J} \rangle'(x) \leq x$ for any x) and that it must be the identity when restricted to $\mathsf{LDB}(\mathbf{J})$.

Let $\Gamma = (\mathbf{V}, \gamma)$ be a view of $\mathbf{D}$. Γ is called an *ideal view* if $\mathbf{V}$ is an ideal subschema of $\mathbf{D}$ and $\gamma = \epsilon\langle \mathbf{D}, \mathbf{V} \rangle$. The set of all ideal views of $\mathbf{D}$ is denoted $\mathsf{IView}(\mathbf{D})$. Note that we have a natural bijective correspondence between $\mathsf{IView}(\mathbf{D})$ and $\mathsf{Ideal}(\mathbf{D})$ via $(\mathbf{V}, \epsilon\langle \mathbf{D}, \mathbf{V} \rangle) \mapsto \mathsf{LDB}(\mathbf{V})$, so that we may reduce the study of ideal views to that of ideals. Also, it is important to note that both $\Gamma_\top(\mathbf{D})$ and $\Gamma_\bot(\mathbf{D})$ are ideal views.

In the theory of lattices, ideals are closed under the join operation. In general, joins need not exist in $\bot$-posets, but when they do, we have that they remain in any ideal from which the elements joined are taken.

1.3.2 Lemma – closure of ideals under least upper bounds *Let J be a complete ideal of $\mathbf{D}$, and let $A \subseteq J$. If $\sup(A)$ exists in $\mathsf{LDB}(\mathbf{D})$, then this supremum is in fact an element of J.*

PROOF: Define $b = \sup(\{y \in J \mid y \leq \sup(A)\})$. Now $b \in J$ by property (ci-ii), and yet $b \leq \sup(A)$. But $c \leq b$ for each $c \in A$, so $\sup(A) \leq b$. Thus $b = \sup(A)$. □

1.3.3 Fixpoints and ideals Let $\Gamma = (\mathbf{V}, \gamma)$ be an ideal view of $\mathbf{D}$. We define $\mathsf{Fixpoint}(\Gamma) = \{x \in \mathsf{LDB}(\mathbf{D}) \mid \gamma'(x) = x\}$. This set is called the *fixpoints* of Γ and also, in harmony with common mathematical usage, the set of *fixpoints* of γ'. The following lemma illustrates the critical importance of fixpoints in the context of ideal views.

1.3.4 Lemma – characterization of ideals in terms of fixpoints *Let $\Gamma = (\mathbf{V}, \gamma)$ be an ideal view of $\mathbf{D}$. Then $\mathsf{Fixpoint}(\Gamma) = \mathsf{LDB}(\mathbf{V})$.*

PROOF: If $x \in \mathsf{LDB}(\mathbf{V})$, then $x = \sup(\{y \in \mathsf{LDB}(\mathbf{V}) \mid y \leq x\})$, and so $\gamma'(x) = x$. On the other hand, if $x \in \mathsf{Fixpoint}(\Gamma)$, then $\gamma'(x) = x$ and since $\gamma'(x) \in \mathsf{LDB}(\mathbf{V})$, we must have $x \in \mathsf{LDB}(\mathbf{V})$. □

1.3.5 Lemma – characterization of ideal view morphisms *Let $\Gamma_1 = (\mathbf{V}_1, \gamma_1)$ and $\Gamma_2 = (\mathbf{V}_2, \gamma_2)$ be ideal views of $\mathbf{D}$. Then:*

(a) *There is a view morphism $\Gamma_1 \to \Gamma_2$ iff $\mathsf{LDB}(\mathbf{V}_2) \subseteq \mathsf{LDB}(\mathbf{V}_1)$.*

(b) *If it exists as a view at all (see 1.1.10), then $\Lambda(\Gamma_1, \Gamma_2)$ is in fact an ideal view of Γ_1, with $\lambda(\Gamma_1, \Gamma_2) = \epsilon\langle \mathbf{V}_1, \mathbf{V}_2 \rangle$.*

(c) *If Γ_1 and Γ_2 are isomorphic, then $\Gamma_1 = \Gamma_2$. In words, isomorphic ideal views are in fact identical.*

PROOF: (a) First assume that the morphism $\lambda(\Gamma_1, \Gamma_2) : \Gamma_1 \to \Gamma_2$ exists, and let $x \in \mathsf{Fixpoint}(\Gamma_2)$, so that $x = \gamma_2'(x) = (\lambda(\Gamma_1, \Gamma_2)' \circ \gamma_1')(x)$. Now $\gamma_1'(x) = \sup(\{y \in \mathsf{LDB}(\mathbf{V}_1) \mid y \leq x\} = \sup(\{y \in \mathsf{LDB}(\mathbf{V}_1) \mid y \leq \gamma_2'(x)\}$, and so $\gamma_1'(x) \in \mathsf{Fixpoint}(\Gamma_2)$ by (ci-i), whence

$\gamma_1'(x) = (\gamma_2' \circ \gamma_1')(x) = (\lambda(\Gamma_1,\Gamma_2)' \circ \gamma_1' \circ \gamma_1')(x) = (\lambda(\Gamma_1,\Gamma_2)' \circ \gamma_1')(x) = \gamma_2'(x) = x$, so that $x \in \mathsf{Fixpoint}(\Gamma_1)$. Thus $\mathsf{LDB}(\mathbf{V}_2) \subseteq \mathsf{LDB}(\mathbf{V}_1)$.

Conversely, suppose that $\mathsf{LDB}(\mathbf{V}_2) \subseteq \mathsf{LDB}(\mathbf{V}_1)$, and let $g : \mathsf{LDB}(\mathbf{V}_1) \to \mathsf{LDB}(\mathbf{V}_2)$ be the mapping defined by $g(x) = \gamma_2'(x)$. Then g is a $\perp$-poset morphism (since it is a restriction of γ_1' and $g \circ \gamma_1' = \gamma_2'$). Hence, in view of the uniqueness result of 1.1.9, g must be the underlying function of $\lambda(\Gamma_1,\Gamma_2)$.

(b) Assume again that the morphism $\lambda(\Gamma_1,\Gamma_2) : \Gamma_1 \to \Gamma_2$ exists, and furthermore let $x \in \mathsf{LDB}(\mathbf{V}_1) \cap \mathsf{Fixpoint}(\Gamma_2)$. Then, by 1.3.4, $x \in \mathsf{Fixpoint}(\Gamma_1)$ as well. Thus $\lambda(\Gamma_1,\Gamma_2)'(x) = (\lambda(\Gamma_1,\Gamma_2)' \circ \gamma_1')(x) = \gamma_2'(x)$, whence $\mathsf{LDB}(\mathbf{V}_2)$ is an ideal of $\mathsf{LDB}(\mathbf{V}_1)$, and so $\Lambda(\Gamma_1,\Gamma_2)$ is an ideal view of $\mathbf{V}_1$, with $\lambda(\Gamma_1,\Gamma_2) = \epsilon\langle \mathbf{V}_1, \mathbf{V}_2\rangle$.

(c) This follows immediately from (b), noting in particular that $\lambda(\Gamma_1,\Gamma_2) = \epsilon\langle \mathbf{V}, \mathbf{V}_2\rangle$ must be the identity. □

1.3.6 Lemma – characterization of ideal intersection *Let J_1 and J_2 be complete ideals of $\mathbf{D}$, and let $x \in \mathsf{LDB}(\mathbf{D})$. Put $x_1 = \sup(\{y \in J_1 \mid y \leq x\})$ and $x_{12} = \sup(\{y \in J_2 \mid y \leq x_1\})$. Then*

(a) $x_{12} \in J_1 \cap J_2$.

(b) $\{y \in J_1 \cap J_2 \mid y \leq x\} = \{y \in J_1 \cap J_2 \mid y \leq x_{12}\}$.

(c) $J_1 \cap J_2$ *is a complete ideal of* $\mathbf{D}$.

PROOF: (a) We have that $x_1 \in J_1$ and $x_{12} \in J_2$ by property (ci-ii). But $x_{12} \in J_1$ as well by property (ci-i), since $x_{12} \leq x_1$.

(b) follows directly from (a).

(c) It is immediate that $J_1 \cap J_2$ satisfies (ci-i). To establish (ci-ii), it suffices to note that x_{12} is already in $J_1 \cap J_2$, so that by (b), $\sup(\{y \in J_1 \cap J_2 \mid y \leq x\}) = \sup(\{y \in J_1 \cap J_2 \mid y \leq x_{12}\}) = x_{12}$. □

1.3.7 Schema and view intersection The previous lemma implies that we may "intersect" ideal subschemata and ideal views as well. If $\mathbf{V}_1$ and $\mathbf{V}_2$ are ideal subschemata of $\mathbf{D}$, we define $\mathbf{V}_1 \cap \mathbf{V}_2$ to be the subschema with $\mathsf{LDB}(\mathbf{V_1} \cap \mathbf{V_2}) = \mathsf{LDB}(\mathbf{V}_1) \cap \mathsf{LDB}(\mathbf{V}_2)$, and the ordering inherited from $\mathbf{D}$. In view of the previous lemma, $\mathbf{V}_1 \cap \mathbf{V}_2$ is an ideal subschema of $\mathbf{D}$. Now if $\Gamma_1 = (\mathbf{V}_1, \gamma_1)$ and $\Gamma_2 = (\mathbf{V}_2, \gamma_2)$ are ideal views of $\mathbf{D}$, we define $\Gamma_1 \cap \Gamma_2 = (\mathbf{V}_1 \cap \mathbf{V}_2, \epsilon\langle \mathbf{D}, \mathbf{V}_1 \cap \mathbf{V}_2\rangle)$.

1.3.8 Relative schemata and views If $\mathbf{J}$ is an ideal subschema of $\mathbf{D}$ and $\Gamma = (\mathbf{V}, \gamma)$ is an ideal view of $\mathbf{D}$, then by 1.3.6(c) and 1.3.7, we know that $\mathbf{J} \cap \mathbf{V}$ is also a subschema of $\mathbf{D}$. But since $\mathsf{LDB}(\mathbf{J} \cap \mathbf{V})$ is in fact a subset of $\mathsf{LDB}(\mathbf{J})$, we may regard $\mathbf{J} \cap \mathbf{V}$ to be a subschema of $\mathbf{J}$, and make the following definitions unambiguously.

(a) If $\Gamma = (\mathbf{V}, \gamma)$ is an ideal view of $\mathbf{D}$, $\Gamma_{|\mathbf{J}}$ denotes the ideal view $(\mathbf{V} \cap \mathbf{J}, \epsilon\langle \mathbf{J}, \mathbf{V} \cap \mathbf{J}\rangle)$ of $\mathbf{J}$.

(b) If Y is a set of ideal views of $\mathbf{D}$, $Y_{|\mathbf{J}}$ denotes $\{\Gamma_{|\mathbf{J}} \mid \Gamma \in Y\}$.

1.3.9 Lemma *Let X be a direct decomposition of $\mathbf{D}$ into ideal views. Then, for any $x \in \mathsf{LDB}(\mathbf{D})$, $x = \sup(\{\gamma_i'(x) \mid i \in I\})$.*

PROOF: Since $\gamma_i'(x) \leq x$ for any i (recall that $\gamma_i' = \epsilon\langle \mathbf{D}, \mathbf{V}_i\rangle'$ and so must be quasi-contracting), it is immediate that x is an upper bound for $\{\gamma_i'(x) \mid i \in I\}$. On the other hand, suppose also that $y \in \mathsf{LDB}(\mathbf{D})$ is such that $\gamma_i'(x) \leq y$ for all $i \in I$. Then $\gamma_i'(x) = (\gamma_i' \circ \gamma_i')(x) \leq \gamma_i'(y)$ for all $i \in I$, whence $x \leq y$, since $\Delta\langle X\rangle$ is an order isomorphism. Hence x is the least upper bound; *i.e.*, $x = \sup(\{\gamma_i'(x) \mid i \in I\})$. □

The following lemma is the critical result. It says that we may always, up to isomorphism, characterize a direct decomposition of a $\perp$-poset schema as a decomposition into ideal views.

1.3.10 Lemma *Suppose that X is a direct decomposition of $\mathbf{D}$, so that the decomposition map $\Delta\langle X\rangle : \mathbf{D} \to \prod_{i\in I} \mathbf{V}_i$ is an isomorphism. Then there is a unique family $\{\mathbf{J}_i \mid i \in I\}$ of ideal subschemata of $\mathbf{D}$ and a family $\{h_i : \mathbf{V}_i \to \mathbf{J}_i \mid i \in I\}$ of isomorphisms, with the inverse of*

$$g \;=\; \mathbf{D} \xrightarrow{\Delta\langle X\rangle} \prod_{i\in I} \mathbf{V}_i \xrightarrow{\prod_{i\in I} h_i} \prod_{i\in I} \mathbf{J}_i$$

being $(x_i)_{i\in I} \mapsto \sup(\{x_i \mid i \in I\})$.

PROOF: For each $k \in I$, put $\mathsf{LDB}(\mathbf{J}_k) = (\Delta\langle X\rangle^{-1})'(\prod_{i\in I} \alpha_i)$, where

$$\alpha_i = \begin{cases} \mathsf{LDB}(\mathbf{V}_i) & \text{if } i = k; \\ \{\perp\} & \text{otherwise.} \end{cases}$$

Now let $h_k : \mathbf{V}_k \to \mathbf{J}_k$ be defined by $x \mapsto (\Delta\langle X\rangle^{-1})'(\delta_{ik}(x))_{i\in I}$, where

$$\delta_{ik}(x) = \begin{cases} x & \text{if } i = k; \\ \perp & \text{otherwise.} \end{cases}$$

It is easily verified that $\mathbf{J}_k$ is an ideal subschema and that h_k is an isomorphism for each $k \in I$. Now for any $x = (x_i)_{i\in I} \in \prod_{i\in I} \mathbf{J}_i$, let $\lambda_k(x)$ be the I-tuple whose k^{th} entry is x_k, and whose other entries are all $\perp$. Then any such x is equal to $\sup(\{\lambda_i(x) \mid i \in I\})$, by the previous lemma. However, g^{-1}, being an isomorphism, preserves all suprema that may exist. Hence $(g^{-1})'(x) = (g^{-1})'(\sup(\{\lambda_i(x) \mid i \in I\})) = \sup(\{(g^{-1})'(\lambda_i(x)) \mid i \in I\}) = \sup(\{x_i \mid i \in I\})$, as was to be shown. Since each $\mathbf{J}_i$ is isomorphic to $\mathbf{V}_i$, the uniqueness follows from 1.3.5(c). □

1.3.11 Theorem – representation of directly complemented views *Every directly complemented view is naturally isomorphic to a unique ideal view.*

PROOF: In the above lemma, we represent the arbitrary complemented view $\Gamma_i = (\mathbf{V}_i, \gamma_i)$ with the ideal view $(\mathbf{J}_i, \epsilon\langle \mathbf{D}, \mathbf{J}_i\rangle)$, via the isomorphism h_i. □

In the examples of 1.1.1 and 1.1.2, we illustrated incompatible decompositions of the example schemata $\mathbf{F}$ and $\mathbf{G}$. We now proceed to establish such incompatibility cannot occur with ideal views. The key is that any direct decomposition of the main schema $\mathbf{D}$ also qualifies as a direct decomposition of any ideal subschema of $\mathbf{D}$. To refine two decompositions into ideal views, we simply decompose all of the views of one using the decomposition specified by the other.

1.3.12 Proposition *Let* $\mathbf{J}$ *be an ideal subschema of* $\mathbf{D}$*, and suppose that* X *is a direct decomposition of* $\mathbf{D}$ *into ideal views. Then* $X_{|\mathbf{J}}$ *is a direct decomposition of the schema* $\mathbf{J}$*.*

PROOF: It suffices to establish that for any $x \in \mathsf{LDB}(\mathbf{D})$, $x \in \mathsf{LDB}(\mathbf{J})$ if and only if $(\forall i \in I)(\gamma_i'(x) \in \mathsf{LDB}(\mathbf{J}) \cap \mathsf{LDB}(\mathbf{V}_i))$. The "$\Rightarrow$" implication is immediate. To show that "$\Leftarrow$" implication, suppose that $x \in \mathsf{LDB}(\mathbf{D})$ with $(\forall i \in I)(\gamma_i'(x) \in \mathsf{LDB}(\mathbf{J}) \cap \mathsf{LDB}(\mathbf{V}_i))$. Now by (ci-ii), $\sup(\{y \in \mathsf{LDB}(\mathbf{J}) \mid y \leq x\} \in \mathsf{LDB}(\mathbf{J})$. But $\{\gamma_i'(x) \mid i \in I\} \subseteq \{y \in \mathsf{LDB}(\mathbf{J}) \mid y \leq x\}$ (since $\gamma_i = \epsilon\langle \mathbf{D}, \mathbf{V}_i\rangle$), and so $x = \sup(\{\gamma_i'(x) \mid i \in I\})$ (see 1.3.9) is in $\mathsf{LDB}(\mathbf{J})$ as well. □

1.3.13 Refinement theorem for decompositions *Let* X *and* Y *be sets of ideal views which are direct decompositions of* $\mathbf{D}$*. Then* $\{\Gamma_x \cap \Gamma_y \mid \Gamma_x \in X$ *and* $\Gamma_y \in Y\}$ *is also a direct decomposition for* $\mathbf{D}$ *into ideal views.*

PROOF: The proof follows directly from the above proposition. We first apply the decomposition X, and then apply the decomposition Y to each subschema of the X decomposition. The resulting views are exactly of the given form. □

1.3.14 Corollary *Let* X *and* Y *be direct decompositions of* $\mathbf{D}$*. Then there is a least decomposition* Z *which is finer than both* X *and* Y*, in the precise sense that every view in* Z *is the meet of a view from* X *and a view from* Y*.* □

1.3.15 Corollary – uniqueness of direct complements *Let* $\{\Gamma_1, \Gamma_2\}$ *and* $\{\Gamma_1, \Gamma_3\}$ *be direct complementary pairs of views of* $\mathbf{D}$*. Then* Γ_2 *and* Γ_3 *are isomorphic. In other words, direct complements are unique up to isomorphism.*

PROOF: By 1.3.11, we know that every directly complemented view of $\mathbf{D}$ is isomorphic to a unique ideal view, so we may assume that the Γ_i's are all ideal. Now we set $X = \{\Gamma_1, \Gamma_2\}$ and $Y = \{\Gamma_1, \Gamma_3\}$, and apply 1.3.13. We get that $\{\Gamma_1 \cap \Gamma_1, \Gamma_1 \cap \Gamma_3, \Gamma_2 \cap \Gamma_1, \Gamma_2 \cap \Gamma_3\}$ is a decomposition. This reduces to $\{\Gamma_1, \Gamma_2 \cap \Gamma_3\}$, since $\Gamma_1 \cap \Gamma_1 = \Gamma_1$ and $\Gamma_1 \cap \Gamma_3 = \Gamma_2 \cap \Gamma_1 = \Gamma_\perp(\mathbf{D})$. Now let $x \in \mathsf{LDB}(\mathbf{V}_3)$; then $(\epsilon\langle \mathbf{D}, \mathbf{V}_1\rangle'(x), \epsilon\langle \mathbf{D}, \mathbf{V}_2\rangle'(x)) = (\perp, x)$. Thus, we must also have $(\epsilon\langle \mathbf{D}, \mathbf{V}_1\rangle'(x),\ \epsilon\langle \mathbf{D}, \mathbf{V}_2 \cap \mathbf{V}_3\rangle'(x)) = (\perp, x)$, whence $\mathsf{LDB}(\mathbf{V}_2 \cap \mathbf{V}_3) = \mathsf{LDB}(\mathbf{V}_3)$; *i.e.*, $\Gamma_2 \cap \Gamma_3 = \Gamma_3$. Thus, the complement is unique up to isomorphism. □

1.3.16 Remark – guaranteed nonuniqueness in the set-based case The above corollary is in striking contrast to Theorem 4.4 of [5], which establishes that minimal complements are never unique within a purely set-based framework, except in the trivial cases of the identity view or the zero view. However, the proof of Bancilhon and Spyratos is for subdirect complements, and requires slight modification to apply to direct complements as well. We sketch the general idea for the direct complement case. Let $\Gamma_1 = (\mathbf{V}_1, \gamma_1)$ and $\Gamma_2 = (\mathbf{V}_2, \gamma_2)$ be complementary views, with neither the identity view nor the zero view. Let $a, b \in \mathsf{LDB}(\mathbf{D})$ with $\gamma_1'(a) = \gamma_1'(b)$. We are then guaranteed that $\gamma_2'(a) \neq \gamma_2'(b)$, else Γ_2 would not be a complement of Γ_1. Define $\gamma_3' : \mathsf{LDB}(\mathbf{D}) \to \mathsf{LDB}(\mathbf{V}_2)$ by the following rule.

$$\gamma_3'(x) = \begin{cases} \gamma_2'(b) & \text{if } x = a; \\ \gamma_2'(a) & \text{if } x = b; \\ \gamma_2'(x) & \text{otherwise.} \end{cases}$$

Now if we drop the requirement that view morphisms satisfy conditions ($\perp$p-i) and ($\perp$p-ii) of 1.1.3, then it is easy to verify that $\Gamma_3 = (\mathbf{V}_3, \gamma_3)$, with $\mathsf{LDB}(\mathbf{V}_2) = \mathsf{LDB}(\mathbf{V}_3)$ will be a "view", which is also a direct complement of Γ_1. Thus, unless it is the zero view or the identity view, if Γ_1 has a direct complement, then it always has another, disregarding ordering constraints. The power of 1.3.15 is in its assertion that such a Γ_3 can never be a $\perp$-poset view.

1.3.17 Notation If $\Gamma = (\mathbf{V}, \gamma)$ is a complemented ideal view of $\mathbf{D}$, we write $\overline{\Gamma} = (\overline{\mathbf{V}}, \overline{\gamma})$ to denote its unique complement in $\mathsf{IView}(\mathbf{D})$.

It is often the case that we wish to extract a decomposition of $\mathbf{D}$ from amongst a set of views. For example, we may speak of a decomposition into projections. It is thus important to have an understanding of the algebraic structure of decompositions within a larger context of a set of views. In the following, we provide some basic results in this direction.

1.3.18 Fully commuting views Given a view $\Gamma = (\mathbf{V}, \gamma)$, the *congruence* of Γ is the set $\mathsf{Congr}(\Gamma) = \{(x, y) \in \mathsf{LDB}(\mathbf{D}) \times \mathsf{LDB}(\mathbf{D}) \mid \gamma'(x) = \gamma'(y)\}$. It is trivial to verify that $\mathsf{Congr}(\Gamma)$ is an equivalence relation on $\mathsf{LDB}(\mathbf{D})$. In the case that Γ is an ideal view, note that the equivalence classes of $\mathsf{Congr}(\Gamma)$ are in bijective correspondence with $\mathsf{LDB}(\mathbf{V})$ via the association $(x \in \mathsf{LDB}(\mathbf{V})) \mapsto \{y \in \mathsf{LDB}(\mathbf{D}) \mid (x, y) \in \mathsf{Congr}(\Gamma)\}$.

Let $\Gamma_1 = (\mathbf{V}_1, \gamma_1)$ and $\Gamma_2 = (\mathbf{V}_2, \gamma_2)$ be any views whatever of $\mathbf{D}$ (not necessary ideal). We say that Γ_1 and Γ_2 are *fully commuting* if $\mathsf{Congr}(\Gamma_1) \circ \mathsf{Congr}(\Gamma_2) = \mathsf{Congr}(\Gamma_2) \circ \mathsf{Congr}(\Gamma_1)$, with "$\circ$" denoting ordinary relational composition. The following result is critical.

1.3.19 Proposition *Let $\Gamma_1 = (\mathbf{V}_1, \gamma_1)$ and $\Gamma_2 = (\mathbf{V}_2, \gamma_2)$ be ideal views of $\mathbf{D}$, and suppose further that $S = \{\Gamma_1, \Gamma_2\}$ be a subdirect decomposition of $\mathbf{D}$ into ideal views. Then S is a direct decomposition if and only if the following two conditions are met.*

(a) $\Gamma_1 \cap \Gamma_2 = \Gamma_\perp(\mathbf{D})$.

(b) Γ_1 *and* Γ_2 *are fully commuting.*

Proof: First of all, let us establish that (a) and (b) together are equivalent to asserting that $\mathsf{Congr}(\Gamma_1) \circ \mathsf{Congr}(\Gamma_2) = \mathsf{Congr}(\Gamma_2) \circ \mathsf{Congr}(\Gamma_1) = \mathsf{LDB}(\mathbf{D}) \times \mathsf{LDB}(\mathbf{D})$. Assume first that (a) and (b) hold, and let $(x, y) \in \mathsf{LDB}(\mathbf{D}) \times \mathsf{LDB}(\mathbf{D})$. Let $(x_1, y_1) = (\epsilon\langle \mathbf{D}, \mathbf{V}_1\rangle'(x), \epsilon\langle \mathbf{D}, \mathbf{V}_1\rangle'(y))$. Then $\epsilon\langle \mathbf{D}, \mathbf{V}_2\rangle'(x_1) = \epsilon\langle \mathbf{D}, \mathbf{V}_2\rangle'(y_1) = \perp$, since since $\Gamma_1 \cap \Gamma_2 = \Gamma_\perp(\mathbf{D})$. Hence $(x_1, \perp), (y_1, \perp) \in \mathsf{Congr}(\Gamma_2)$. Since $(x, x_1), (y, y_1) \in \mathsf{Congr}(\Gamma_1)$ by construction, it follows that $(x, y) \in \mathsf{Congr}(\Gamma_1) \circ \mathsf{Congr}(\Gamma_2) \circ \mathsf{Congr}(\Gamma_2) \circ \mathsf{Congr}(\Gamma_1) = \mathsf{Congr}(\Gamma_1) \circ \mathsf{Congr}(\Gamma_2)$; *i.e.*, these two congruences commute and their composition is $\mathsf{LDB}(\mathbf{D}) \times \mathsf{LDB}(\mathbf{D})$.

Conversely, assume that $\mathsf{Congr}(\Gamma_1) \circ \mathsf{Congr}(\Gamma_2) = \mathsf{Congr}(\Gamma_2) \circ \mathsf{Congr}(\Gamma_1) = \mathsf{LDB}(\mathbf{D}) \times \mathsf{LDB}(\mathbf{D})$. Let $x \in \mathsf{LDB}(\mathbf{V}_1) \cap \mathsf{LDB}(\mathbf{V}_2)$. Then, in particular, $(x, \perp) \in \mathsf{Congr}(\Gamma_1) \circ \mathsf{Congr}(\Gamma_2)$, so there is a $y \in \mathsf{LDB}(\mathbf{D})$ such that $(x, y) \in \mathsf{Congr}(\Gamma_1)$ and $(y, \perp) \in \mathsf{Congr}(\Gamma_2)$. Now $x \leq y$, since $y = \epsilon\langle \mathbf{D}, \mathbf{V}_1\rangle'(x)$, and so $\epsilon\langle \mathbf{D}, \mathbf{V}_2\rangle'(x) \leq \epsilon\langle \mathbf{D}, \mathbf{V}_2\rangle'(y) = \perp$, *i.e.*, $(x, \perp) \in \mathsf{Congr}(\Gamma_2)$. But since $x \in \mathsf{LDB}(\mathbf{V}_2)$, we must have $x = \perp$. Thus $\Gamma_1 \cap \Gamma_2 = \Gamma_\perp(\mathbf{D})$.

Note further that S is a subdirect decomposition if and only if $\mathsf{Congr}(\Gamma_1) \cap \mathsf{Congr}(\Gamma_2) = \{(x, x) \mid x \in \mathsf{LDB}(\mathbf{D})\}$. Armed with these facts, the key is to use the well-known result on direct decompositions of algebras which states that $\Delta\langle S\rangle'$ is a bijection if and only if Γ_1 and Γ_2

are fully commuting, $\mathsf{Congr}(\Gamma_1) \circ \mathsf{Congr}(\Gamma_2) = \mathsf{LDB}(\mathbf{D}) \times \mathsf{LDB}(\mathbf{D})$, and $\mathsf{Congr}(\Gamma_1) \cap \mathsf{Congr}(\Gamma_2) = \{(x, x) \mid x \in \mathsf{LDB}(\mathbf{D})\}$ [8, Cor. 2, Chap. 4]. But since we are assuming that S is a subdirect decomposition, we know that $\Delta\langle S\rangle$ is a section. Hence surjectivity is sufficient to guarantee that it is an isomorphism, by 1.1.4(c). □

1.3.20 The Bounded Weak Partial Lattice of Ideal Views There is a natural partial order structure on $\mathsf{IView}(\mathbf{D})$ given by $\Gamma_1 \leq \Gamma_2$ if and only if $\Gamma_1 \cap \Gamma_2 = \Gamma_1$. In terms of ideals, if $\Gamma_1 = (\mathbf{V}_1, \gamma_1)$ and $\Gamma_2 = (\mathbf{V}_2, \gamma_2)$, this is equivalent to $\mathsf{LDB}(\mathbf{V}_1) \subseteq \mathsf{LDB}(\mathbf{V}_2)$. (See 1.3.5(a).) This order has the greatest element $\Gamma_\top(\mathbf{D})$ and the least element $\Gamma_\bot(\mathbf{D})$. Our goal is to create a lattice-like structure based upon this ordering which will enable us to represent the decompositions of $\mathbf{D}$ as suitable substructures. The following definitions are the appropriate ones.

We define two partial operations on $\mathsf{IView}(\mathbf{D})$. The *meet* operation $\wedge : \mathsf{IView}(\mathbf{D}) \times \mathsf{IView}(\mathbf{D}) \to \mathsf{IView}(\mathbf{D})$ is given by

$$\Gamma_1 \wedge \Gamma_2 = \begin{cases} \Gamma_1 \cap \Gamma_2 & \text{if } \Gamma_1 \text{ and } \Gamma_2 \text{ are fully commuting} \\ \text{undefined} & \text{otherwise} \end{cases}$$

The *join* operation $\vee : \mathsf{IView}(\mathbf{D}) \times \mathsf{IView}(\mathbf{D}) \to \mathsf{IView}(\mathbf{D})$ is given by

$$\Gamma_1 \vee \Gamma_2 = \begin{cases} \sup(\{\Gamma_1, \Gamma_2\}) & \text{if this supremum exists in the underlying partial order} \\ \text{undefined} & \text{otherwise} \end{cases}$$

The structure $(\mathsf{IView}(\mathbf{D}), \vee, \wedge, \Gamma_\top(\mathbf{D}), \mathbf{\Gamma}_\bot(\mathbf{D}))$ constitutes what is known as a *bounded weak partial lattice*, and is denoted $\mathbf{IView}(\mathbf{D})$. For a complete discussion of the definition and terminology, consult [25, Chap. I, Sec. 5]. For our purposes, it is sufficient to know that if we extract a substructure in which the operations of join and meet are total, then we get a lattice.

A *Boolean subalgebra* of $\mathbf{IView}(\mathbf{D})$ is a Boolean algebra[2] $\mathbf{B} = (B, \vee_{\mathbf{B}}, \wedge_{\mathbf{B}}, \overline{(-)}, \top_{\mathbf{B}}, \bot_{\mathbf{B}})$ with $B \subseteq \mathsf{IView}(\mathbf{D})$, $\top_{\mathbf{B}} = \Gamma_\top(\mathbf{D})$, $\bot_{\mathbf{D}} = \Gamma_\bot(\mathbf{D})$, and $\vee_{\mathbf{B}}$ and $\wedge_{\mathbf{B}}$ the respective restrictions of $\vee$ and $\wedge$ to $B \times B$. The complementation operation $\overline{(-)}$ is necessarily induced by the underlying lattice structure [25, Ch. I, Sec. 6, Lem. 1].

1.3.21 Counterexample It is not the case that the congruences of any two ideal are fully commuting, so the meet operation of $\mathbf{IView}(\mathbf{D})$ is indeed partial. For a specific counterexample, let $S = \{a, b, c\}$ and let $\mathbf{P}$ be the schema with $\mathsf{LDB}(\mathbf{P})$ the set of all subsets of S except $\{a, b\}$, ordered by set inclusion. Think of $\mathbf{P}$ as governed by the single "dependency" $(\forall M \in \mathsf{LDB}(\mathbf{P}))(((a \in M) \wedge (b \in M)) \Rightarrow (c \in M))$. Define $\Gamma_1 = (\mathbf{V}_1, \gamma_1)$ to be the ideal view with $\mathsf{LDB}(\mathbf{V}_1) = \{\emptyset, \{a\}\}$, and define $\Gamma_2 = (\mathbf{V}_2, \gamma_2)$ to be the ideal view with $\mathsf{LDB}(\mathbf{V}_2) = \{\emptyset, \{b\}, \{c\}, \{b, c\}\}$. The congruences of Γ_1 and Γ_2 do not commute, as $(\{b\}, \{a, c\}) \in (\mathsf{Congr}(\Gamma_1) \circ \mathsf{Congr}(\Gamma_2)) \setminus (\mathsf{Congr}(\Gamma_2) \circ \mathsf{Congr}(\Gamma_1))$. Fortunately, we have the following.

[2]We no not explicitly disallow the trivial case of a one-element Boolean algebra, but this can only occur if $\mathbf{D}$ is the zero schema.

1.3.22 Lemma *Let X be a direct decomposition of $\mathbf{D}$. Then any two elements of X are fully commuting.*

PROOF: Let $\Gamma_j \in X$. It is immediate that $\mathsf{Congr}(\overline{\Gamma_j}) = \bigcap\{\mathsf{Congr}(\Gamma_i) \mid i \in I \setminus \{j\}\}$. By 1.3.16, we have that $\mathsf{Congr}(\Gamma_j)$ and $\mathsf{Congr}(\overline{\Gamma_j})$ are fully commuting with $\mathsf{Congr}(\Gamma_j) \circ \mathsf{Congr}(\overline{\Gamma_j}) = \mathsf{LDB}(\mathbf{D}) \times \mathsf{LDB}(\mathbf{D})$. Since for any $i \in I \setminus \{j\}$ we have $\mathsf{Congr}(\overline{\Gamma_j}) \subseteq \mathsf{Congr}(\Gamma_i)$, it follows that $\mathsf{Congr}(\Gamma_j) \circ \mathsf{Congr}(\Gamma_i) = \mathsf{Congr}(\Gamma_i) \circ \mathsf{Congr}(\Gamma_j) = \mathsf{LDB}(\mathbf{D}) \times \mathsf{LDB}(\mathbf{D})$ as well. □

1.3.23 Main decomposition theorem *The directly complemented ideal views of $\mathbf{D}$ form a Boolean subalgebra of the bounded weak partial lattice $\mathbf{IView}(\mathbf{D})$. In this Boolean algebra, the (Boolean) complement is the direct complement, and the decompositions of $\mathbf{D}$ are in bijective correspondence with the finite Boolean subalgebras of $\mathbf{IView}(\mathbf{D})$.*

PROOF: Let $\Gamma_1 = (\mathbf{V}_1, \gamma_1)$ and $\Gamma_2 = (\mathbf{V}_2, \gamma_2)$ each be directly complemented ideal views. Define Γ_{12} to be $\Gamma_1 \cap \Gamma_2$, $\Gamma_{1\overline{2}}$ to be $\Gamma_1 \cap \overline{\Gamma_2}$, $\Gamma_{\overline{1}2}$ to be $\overline{\Gamma_1} \cap \Gamma_2$, and $\Gamma_{\overline{12}}$ to be $\overline{\Gamma_1} \cap \overline{\Gamma_2}$. Then $\{\Gamma_{12}, \Gamma_{1\overline{2}}, \Gamma_{\overline{1}2}, \Gamma_{\overline{12}}\}$ forms a direct decomposition of $\mathbf{D}$ by 1.3.13, and any pair from this set is fully commuting, by the previous lemma.

First let us address the existence of the meet of two directly complemented ideal views. It is immediate that $\mathsf{Congr}(\Gamma_1) = \mathsf{Congr}(\Gamma_{12}) \cap \mathsf{Congr}(\Gamma_{1\overline{2}})$, and $\mathsf{Congr}(\Gamma_2) = \mathsf{Congr}(\Gamma_{12}) \cap \mathsf{Congr}(\Gamma_{\overline{1}2})$, and so $\mathsf{Congr}(\Gamma_1) \circ \mathsf{Congr}(\Gamma_2) = \mathsf{Congr}(\Gamma_{12})$. Similarly, $\mathsf{Congr}(\Gamma_2) \circ \mathsf{Congr}(\Gamma_1) = \mathsf{Congr}(\Gamma_{12})$. Thus, Γ_1 and Γ_2 are fully commuting, and hence $\Gamma_1 \wedge \Gamma_2$ exists in $\mathbf{IView}(\mathbf{D})$ as $\Gamma_1 \cap \Gamma_2$.

Next, we turn to the issue of existence of the join $\Gamma_1 \vee \Gamma_2$ of two directly complemented ideal views Γ_1 and Γ_2. Now if $\Gamma_1 \vee \Gamma_2$ is to exist as a complemented element of $\mathbf{IView}(\mathbf{D})$, then de Morgan's classic identity [25, Ch. I, Sec. 6, Lemma 3] tells us that it must be $\overline{\overline{\Gamma_1} \wedge \overline{\Gamma_2}} = \overline{\Gamma_{\overline{12}}}$. Let us now directly construct this view, making use of the fact that we already know that $\{\Gamma_{12}, \Gamma_{1\overline{2}}, \Gamma_{\overline{1}2}, \Gamma_{\overline{12}}\}$ is a direct decomposition of $\mathbf{D}$. Extending the notation of the previous paragraph, we write $\Gamma_{12} = (\mathbf{V}_{12}, \gamma_{12})$, $\Gamma_{1\overline{2}} = (\mathbf{V}_{1\overline{2}}, \gamma_{1\overline{2}})$, $\Gamma_{\overline{1}2} = (\mathbf{V}_{\overline{1}2}, \gamma_{\overline{1}2})$, and $\Gamma_{\overline{12}} = (\mathbf{V}_{\overline{12}}, \gamma_{\overline{12}})$. Define $\Gamma_3 = (\mathbf{V}_3, \gamma_3)$ by $\mathsf{LDB}(\mathbf{V}_3) = \mathsf{LDB}(\mathbf{V}_{12}) \times \mathsf{LDB}(\mathbf{V}_{1\overline{2}}) \times \mathsf{LDB}(\mathbf{V}_{\overline{1}2})$ and $\gamma_3' : \mathsf{LDB}(\mathbf{D}) \to \mathsf{LDB}(\mathbf{V}_3)$ by $x \mapsto (\gamma_{12}'(x), \gamma_{1\overline{2}}'(x), \gamma_{\overline{1}2}'(x))$. It is immediate that Γ_3 is a view; indeed $\gamma_3 = g \circ \Delta\langle\{\Gamma_{12}, \Gamma_{1\overline{2}}, \Gamma_{\overline{1}2}, \Gamma_{\overline{12}}\}\rangle$, with $g' : \mathsf{LDB}(\mathbf{V}_{12}) \times \mathsf{LDB}(\mathbf{V}_{1\overline{2}}) \times \mathsf{LDB}(\mathbf{V}_{\overline{1}2}) \times \mathsf{LDB}(\mathbf{V}_{\overline{12}}) \to \mathsf{LDB}(\mathbf{V}_{12}) \times \mathsf{LDB}(\mathbf{V}_{1\overline{2}}) \times \mathsf{LDB}(\mathbf{V}_{\overline{1}2})$ given on elements by $(x_{12}, x_{1\overline{2}}, x_{\overline{1}2}, x_{\overline{12}}) \mapsto (x_{12}, x_{1\overline{2}}, x_{\overline{1}2})$. In words, g throws away the $\Gamma_{\overline{12}}$ component, and preserves the other three. Furthermore, $\{\Gamma_{\overline{12}}, \Gamma_3\}$ is a direct complementary pair, since $\Delta\langle\{\Gamma_1, \Gamma_3\}\rangle = \Delta\langle\{\Gamma_{12}, \Gamma_{1\overline{2}}, \Gamma_{\overline{1}2}, \Gamma_{\overline{12}}\}\rangle$. But now, by 1.3.11, Γ_3 is isomorphic to an ideal view, which must be $\overline{\Gamma_{\overline{12}}}$.

To complete the proof that $\overline{\Gamma_{\overline{12}}}$ is indeed $\Gamma_1 \vee \Gamma_2$, we must show that it is the smallest ideal view which is larger than both Γ_1 and Γ_2, using the definition of join in 1.3.20. Now $\overline{\Gamma_{\overline{12}}} = \overline{\overline{\Gamma_1} \cap \overline{\Gamma_2}}$ is a directly complemented ideal view, as just established. We claim that it is in fact $\Gamma_1 \vee \Gamma_2$. To establish this, let $\Gamma_5 = (\mathbf{V}_5, \gamma_5)$ be any ideal view with the properties $\Gamma_1 \leq \Gamma_5$, $\Gamma_2 \leq \Gamma_5$, and $\Gamma_5 \leq \overline{\Gamma_{\overline{12}}}$. We use the notation $\overline{\Gamma_{\overline{12}}} = (\overline{\mathsf{LDB}(\mathbf{V}_{\overline{12}})}, \overline{\gamma_{\overline{12}}})$, and let $x \in \overline{\mathsf{LDB}(\mathbf{V}_{\overline{12}})}$. Now set $(x_{12}, x_{1\overline{2}}, x_{\overline{1}2}, x_{\overline{12}}) = \Delta\langle\{\Gamma_{12}, \Gamma_{1\overline{2}}, \Gamma_{\overline{1}2}, \Gamma_{\overline{12}}\}\rangle'(x) \in \mathsf{LDB}(\mathbf{V}_{12}) \times \mathsf{LDB}(\mathbf{V}_{1\overline{2}}) \times \mathsf{LDB}(\mathbf{V}_{\overline{1}2}) \times \mathsf{LDB}(\mathbf{V}_{\overline{12}})$ to be the decomposition of x by the four-element direct decomposition $\{\Gamma_{12}, \Gamma_{1\overline{2}}, \Gamma_{\overline{1}2}, \Gamma_{\overline{12}}\}$ of $\mathbf{D}$. It is immediate that $x_{\overline{12}} = \bot$, since $x \in \overline{\mathsf{LDB}(\mathbf{V}_{\overline{12}})}$. But since $\Gamma_{12}, \Gamma_{1\overline{2}} \leq \Gamma_1 \leq \Gamma_5$ and $\Gamma_{\overline{1}2} \leq \Gamma_2 \leq \Gamma_5$, we have that $\{x_{12}, x_{1\overline{2}}, x_{\overline{1}2}\} \subseteq \mathsf{LDB}(\mathbf{V}_5)$. But $x = \sup(\{x_{12}, x_{1\overline{2}}, x_{\overline{1}2}, x_{\overline{12}}) = \sup(\{x_{12}, x_{1\overline{2}}, x_{\overline{1}2}, \bot\}) = \sup(\{x_{12}, x_{1\overline{2}}, x_{\overline{1}2}\}) \in \mathsf{LDB}(\mathbf{V}_5)\}$ by 1.3.9 and 1.3.2. Thus $\overline{\Gamma_{\overline{12}}} \leq \Gamma_5$, and so these two views must be equal. Hence, $\overline{\Gamma_{\overline{12}}}$ is the least upper bound of Γ_1 and Γ_2, and so is $\Gamma_1 \vee \Gamma_2$, by definition.

Finally, we must address the issue of complements. We have already established that complements of meets of directly complemented ideal views exist, and the existence of complements for joins then follows from de Morgan's identity. That the Boolean complement is the direct complement is immediate from the definitions.

Conversely, let Γ_1 and Γ_2 be complementary elements in $\mathbf{IView}(\mathbf{D})$ (in the sense that $\Gamma_1 \wedge \Gamma_2 = \Gamma_\perp(\mathbf{D})$ and $\Gamma_1 \vee \Gamma_2 = \Gamma_\top(\mathbf{D})$). Then it is immediate from the definition of $\wedge$ in $\mathbf{IView}(\mathbf{D})$ and from 1.3.17 that $\{\Gamma_1, \Gamma_2\}$ forms a direct complementary pair. Hence any Boolean subalgebra of $\mathbf{IView}(\mathbf{D})$ must contain only directly complemented views, whence any Boolean subalgebra of $\mathbf{IView}(\mathbf{D})$ must be itself a subalgebra of the Boolean algebra of all complemented views.

We have now established that the directly complemented ideal views of $\mathbf{D}$ form a Boolean subalgebra of $\mathbf{IView}(\mathbf{D})$, and that the algebra complement and direct complement coincide. To complete the proof, we must show that the decompositions of $\mathbf{D}$ are in bijective correspondence with the finite Boolean subalgebras of $\mathsf{IView}(\mathbf{D})$. If X is a decomposition, then the elements of X are independent, by 1.3.17 and 1.3.20. Hence they surely generate a finite Boolean algebra in which they are the atoms. Conversely, let $\mathbf{B}$ is a finite Boolean subalgebra of the algebra of all complemented views, with atoms[3] X. Then, for any two element partition $\{X_1, X_2\}$ of X, $\{\bigvee X_1, \bigvee X_2\}$ will be a complementary pair in $\mathbf{IView}(\mathbf{D})$. However, since view complement and algebra complement are the same, it follows that this pair is also a direct decomposition of $\mathbf{D}$. Finally, repeated application of 1.3.11 then assures us that X will also be a decomposition. Thus, the decompositions of $\mathbf{D}$ are in bijective correspondence with the finite Boolean subalgebras. □

1.3.24 Notation We let $\mathsf{DCIView}(\mathbf{D})$ denote the set of all directly complemented ideal views of $\mathbf{D}$, and we let $\mathbf{DCIView}(\mathbf{D})$ denote the Boolean algebra whose underlying set is $\mathsf{DCIView}(\mathbf{D})$, with operations are those inherited from $\mathbf{IView}(\mathbf{D})$.

We next show that any Boolean algebra (up to isomorphism) can arise as $\mathsf{DCIView}(\mathbf{D})$ with appropriate choice of $\mathbf{D}$, so that it is impossible to assert anything more about $\mathsf{DCIView}(\mathbf{D})$ without making some restrictions on $\mathbf{D}$. The easiest way to see this is to take an example in which $\mathsf{LDB}(\mathbf{D})$ has the structure of a Boolean algebra. It is then the case that the decompositions of $\mathbf{D}$ are in bijective correspondence with the Boolean subalgebras of $\mathsf{LDB}(\mathbf{D})$.

1.3.25 Proposition *Let $\mathbf{D}$ be a poset which is also a Boolean algebra. Then the ideal views of $\mathbf{D}$ are all defined by principal ideals, and they are in bijective correspondence with the elements of $\mathsf{LDB}(\mathbf{D})$, via the association $(x \in \mathsf{LDB}(\mathbf{D})) \mapsto \langle x \rangle$. This bijection furthermore defines a Boolean algebra isomorphism.*

PROOF: Let $x \in \mathsf{LDB}(\mathbf{D})$. Let $\top_{\mathbf{D}}$ denote the greatest element of $\mathbf{D}$. If J is an ideal of $\mathbf{D}$, then by (ci-ii) $\sup(\{y \in J \mid y \leq \top_{\mathbf{D}}\}) \in J$, and this must be the greatest element in J. Thus, by (ci-i), $J = \{x \in \mathsf{LDB}(\mathbf{D}) \mid x \leq \sup\{y \in J \mid y \leq \top_{\mathbf{D}}\}\}$; that is, J is the principal ideal $\langle \sup(\{y \in J \mid y \leq \top_{\mathbf{D}}\}) \rangle$. Upon translating from ideals to ideal views, we have the required bijection. □

[3]Recall that x is an *atom* in a Boolean algebra if for any y with $y \leq x$, either $y = x$ or else y is the least element.

Decompositions of database schemata are naturally regarded to be into a finite number of views. To achieve a finite ultimate decomposition in a particular application, we must establish that the complemented ideal views are finite in number. The following definition recaptures this.

1.3.26 Definition A *finite decomposition framework* is a sublattice $\mathbf{L}$ of $\mathsf{IView}(\mathbf{D})$ with the greatest element $\Gamma_\top(\mathbf{D})$ and least element $\Gamma_\bot(\mathbf{D})$, such that the set of all complemented elements of $\mathbf{L}$ is finite (and so $\mathsf{DCIView}(\mathbf{D})$ forms a finite Boolean algebra).

1.3.27 Proposition *Let* $\mathbf{L}$ *be a finite decomposition framework. Then* $\mathbf{D}$ *has a unique ultimate direct decomposition* Y *within* $\mathbf{L}$*, with* Y *consisting of precisely the atoms of* $\mathbf{L}$*.*

PROOF: It is immediate that the set of all *complemented* atoms of $\mathbf{L}$ forms a direct decomposition of $\mathbf{D}$ which cannot further be refined within $\mathbf{L}$. Now let $\Gamma = (\mathbf{V}, \gamma)$ be any atom of $\mathbf{L}$. Using 1.3.12, we may decompose $\mathbf{V}$ using the atoms of $\mathbf{L}$. But for each such atom $\Gamma_\alpha = (\mathbf{V}_\alpha, \gamma_\alpha)$, we must have that $\mathsf{LDB}(\mathbf{V}_\alpha) \cap \mathsf{LDB}(\mathbf{V})$ is either $\mathsf{LDB}(\mathbf{V}_\alpha)$ or $\{\bot\}$, else $\mathsf{LDB}(\mathbf{V}_\alpha)$ would not be an atom of $\mathbf{L}$. Hence Γ must itself be one of the members of Y; in other words, all atoms of $\mathbf{L}$ are complemented. □

2. Application to Relational Schema Decomposition

A good general decomposition theory for database schemata should at least recapture the most important specific results. In this part of the paper, we examine the most fundamental ways in which the theory of Part 1 interacts with the relational model, and we show how to completely recapture join-based decomposition.

2.1 The General Structure of Relational Decomposition

The traditional framework for relational schema decomposition, being based upon the idea of disjoint domains, does not provide the formal framework necessary to adequately deal with null values, which are essential for projection-based decomposition into independent components. Rather than augment the traditional framework in an ad hoc fashion, we work directly with a framework in which the domains form a finite Boolean algebra. This idea has already been employed in some logic-based approaches to relational database theory, such as [45]. This not only provides a simple means of formalizing null values, but it also allows us to model horizontal decompositions based upon type, as first suggested by Smith [46].

2.1.1 Type algebras

(a) A *type algebra* is a triple $\mathcal{T} = (\mathbf{T}, \mathbf{K}, \mathbf{A})$, where:

(i) $\mathbf{T}$ is a finite set of unary relation symbols, called the *types*.

(ii) $\mathbf{K}$ is a finite set of constant symbols, called the *names*. For convenience, it is always assumed that $\mathbf{T} \cap \mathbf{K} = \emptyset$.

(iii) $\mathbf{A}$ is a theory in the language of $\mathsf{Lang}(\mathcal{T})$ of $\mathcal{T}$, which is the first-order language with equality whose (other) relational symbols are those of $\mathbf{T}$, whose constant symbols are those of $\mathbf{K}$, and which has no other function symbols.

(iv) For each $\tau_1, \tau_2 \in \mathbf{T}$, there is a $\tau_3 \in \mathbf{T}$ such that $\mathbf{A} \models (\forall x)(\tau_3(x) \Leftrightarrow \tau_1(x) \vee \tau_2(x))$. τ_3 is denoted by $\tau_1 \vee \tau_2$ and is called the *union* (or *join*) of τ_1 and τ_2.

(v) For each $\tau_1, \tau_2 \in \mathbf{T}$, there is a $\tau_3 \in \mathbf{T}$ such that $\mathbf{A} \models (\forall x)(\tau_3(x) \Leftrightarrow \tau_1(x) \wedge \tau_2(x))$. τ_3 is denoted by $\tau_1 \wedge \tau_2$ and is called the *intersection* (or *meet*) of τ_1 and τ_2.

(vi) For each $\tau \in \mathbf{T}$, there is a $\bar{\tau} \in \mathbf{T}$ such that $\mathbf{A} \models (\forall x)(\bar{\tau}(x) \Leftrightarrow \neg\tau(x))$. $\bar{\tau}$ is called the *complement* of τ.

(vii) There is a type $\tau_\top \in \mathbf{T}$ such that $\mathbf{A} \models (\forall x)(\tau_\top(x))$. $\tau_\top$ is called the *universal type*. The complement of $\tau_\top$ is called the *empty type*, and is denoted by $\tau_\bot$.

(viii) For each $a \in \mathbf{K}$ and $\tau \in \mathbf{T}$, either $\mathbf{A} \models \tau(a)$ or else $\mathbf{A} \models \bar{\tau}(a)$.

(b) Let $\tau_1, \tau_2 \in \mathbf{T}$ in the type algebra $\mathcal{T}$.

(i) We write $\tau_1 \leq \tau_2$ if $(\forall x)(\tau_1(x) \Rightarrow \tau_2(x)) \in \mathbf{A}$.

(ii) We say that τ_1 and τ_2 are *equivalent* (written $\tau_1 = \tau_2$) if $\tau_1 \leq \tau_2$ and $\tau_2 \leq \tau_1$.

(c) Let $\tau \in \mathbf{T}$. τ is *atomic* if for any type $\tau_1 \in \mathbf{T}$ for which $\tau_1 \leq \tau$, either $\tau_1 = \tau$ or else $\tau_1 = \tau_\bot$. The set of all atomic types of $\mathcal{T}$ is denoted by $\mathsf{Atoms}(\mathcal{T})$.

(d) For $a \in \mathbf{K}$, the *base type* of a is the least type $\tau \in \mathbf{T}$ such that $\mathbf{A} \models \tau(a)$, and is denoted $\mathsf{BaseType}(a)$. More generally, for any type υ, we say that say that a is of type υ if $\mathbf{A} \models \upsilon(a)$. Clearly, this is the case if and only if $\mathsf{BaseType}(a) \leq \upsilon$.

Equivalent types are just different names for the same predicate. It is clear that conditions (iv)-(vii) endow equivalence classes of elements of $\mathbf{T}$ with the structure of a finite Boolean algebra. Since no confusion can result from the renaming, we shall not develop a special notation for equivalence classes, but rather just regard $\mathbf{T}$ as a Boolean algebra, with the understanding that a type may have more than one name. $\tau_\top$ is the greatest element of the algebra; its complement is the empty type $\tau_\bot$. Note that the atomic types in the sense of (c) above are precisely those types which are atoms in this underlying Boolean algebra.

2.1.2 Type assignments Let $\mathcal{T} = (\mathbf{T}, \mathbf{K}, \mathbf{A})$ be a type algebra. A *type assignment* for $\mathcal{T}$ is any model μ of $\mathbf{A}$. More precisely, μ is defined by a set $\mathcal{D}(\mu)$, together with an assignment to each $\tau \in \mathbf{T}$ of a subset (= unary relation) $\mathsf{Dom}_\mu(\tau)$ of $\mathcal{D}(\mu)$, and to each $k \in \mathbf{K}$ an element $k^\mu \in \mathcal{D}(\mu)$, in such a way that the axioms $\mathbf{A}$ are satisfied. In keeping with the traditional approach, $\mathsf{Dom}_\mu(\tau)$ is called the *domain* of τ, and $\mathcal{D}(\mu)$ is called the *universe* of μ. We extend the notion of base type to domain elements by defining, for each $x \in \mathcal{D}(\mu)$, $\mathsf{BaseType}(x, \mu)$ to be the unique atomic type for which $x \in \mathsf{Dom}_\mu(\tau)$. Note that under this setup, $\{\mathsf{Dom}_\mu(\tau) \mid \tau \in \mathbf{T}\}$ truly becomes a Boolean algebra under the usual set-theoretic operations of union, intersection, and complement.

Type assignments are the analog of the assignments of domains to attributes in the more conventional approach. As such, we would like them to be uniquely defined by the axioms of $\mathcal{T}$. Unfortunately, this is not possible within first-order means unless we axiomatize for each

$\tau \in \mathbf{T}$ a fixed finite upper bound on the size of each $\mathsf{Dom}_\tau(\mu)$, independently of μ. The limited compromises which are possible are outside of the scope of this work. Here, we will merely ensure that the underlying type assignment is not changed by database mappings, so that the base schema and all views refer to the same domain.

2.1.3 Convention Throughout the rest of this paper, unless specifically stated to the contrary, we assume that there is a fixed type algebra $\mathcal{T} = (\mathbf{T}, \mathbf{K}, \mathbf{A})$, as well as a fixed but arbitrary type assignment μ. In any results, we must ensure that the statements hold regardless of the choice of μ.

2.1.4 Relational schemata and instances A *relational schema* $\mathbf{D} = (\mathsf{Rel}(\mathbf{D}), \mathsf{Con}(\mathbf{D}))$ over $\mathcal{T} = (\mathbf{T}, \mathbf{K}, \mathbf{A})$ is a pair such that $\mathsf{Rel}(\mathbf{D})$ is a finite set of relational symbols and $\mathsf{Con}(\mathbf{D})$ is a set of sentences in the first-order language with equality whose nonlogical symbols include precisely the nonlogical symbols of $\mathcal{T}$ plus the relation symbols in $\mathsf{Rel}(\mathbf{D})$. This language is called the *language of* $\mathbf{D}$, and is denoted $\mathsf{Lang}(\mathbf{D})$. We always assume that $\mathsf{Con}(\mathbf{D}) \models \mathbf{A}$. A *basis* for $\mathsf{Con}(\mathbf{D})$ is any set Σ of sentences with $\Sigma \cup \mathbf{A} \models \mathsf{Con}(\mathbf{D})$. The schema $\mathbf{D}$ is *finitely axiomatizable* if $\mathsf{Con}(\mathbf{D})$ has a finite basis. The arity (number of arguments) of a symbol $R \in \mathsf{Rel}(\mathbf{D})$ is denoted by $\mathsf{Ar}(R)$.

A *legal database* (or *instance*) of the schema $\mathbf{D}$ is just a model of $\mathsf{Con}(\mathbf{D})$; that is, a structure which satisfies all of the constraints. In keeping with the notation of the previous sections, the set of all legal databases of $\mathbf{D}$ which agree with μ is denoted $\mathsf{LDB}(\mathbf{D}, \mu)$. Occasionally, we will need to speak of databases which are not legal. $\mathsf{DB}(\mathbf{D}, \mu)$ denotes those structures which are legal databases of $(\mathsf{Rel}(\mathbf{D}), \mathbf{A})$ and which agree with μ. In other words, this latter set ignores the constraints of the schema, except for those of the underlying type algebra.

If $M \in \mathsf{DB}(\mathbf{D}, \mu)$, the associated relation for $R \in \mathsf{Rel}(\mathbf{D})$ is denoted R^M. The operations of intersection and union are defined relation-wise on $\mathsf{DB}(\mathbf{D}, \mu)$. Thus, for nonempty $Y \subseteq \mathsf{DB}(\mathbf{D}, \mu)$ and $R \in \mathsf{Rel}(\mathbf{D})$, $R^{\cap Y} = \bigcap\{R^M \mid M \in Y\}$, and $R^{\cup Y} = \bigcup\{R^M \mid M \in Y\}$.

2.1.5 The rôle of type assignments in our approach In the traditional approach to relational database theory [39], attributes play a central rôle in that they define the admissible domains for columns of relations. In our approach, constraints defining admissible types for columns take over this rôle. An example will make this clear. Suppose that R is a ternary relation symbol, and suppose that the attributes for columns 1, 2, and 3 are A, B, and C, respectively. In the usual notation, we would write $R[ABC]$. A legal relation of R would be constrained to consist of triples from $\mathsf{dom}(A) \times \mathsf{dom}(B) \times \mathsf{dom}(C)$, where the dom's are preassigned attribute domains. In our approach, such constraints are expressed directly using types. To each attribute assign a type; call these τ_A, τ_B, and τ_C, respectively. Then the typing constraint is expressed by

$$(\forall x)(\forall y)(\forall z)(R(x, y, z) \Rightarrow \tau_A(x) \wedge \tau_B(y) \wedge \tau_C(z))$$

and this sentence becomes member of $\mathsf{Con}(\mathbf{D})$. We may still use notation like $R[ABC]$, with the understanding that it is an abbreviation for the existence of underlying types τ_A, τ_B, τ_C, as well as the above first-order constraint.

2.1.6 Orderings on $\mathsf{LDB}(\mathbf{D}, \mu)$ In order to apply the theory developed in the first part of this paper, we must assign an ordering to the database states in $\mathsf{LDB}(\mathbf{D}, \mu)$. There are (at least) two such ordering of importance in the relational theory. The first is the natural inclusion ordering, and simply orders the states on the basis of relation-by-relation inclusion. The second is called the null-augmented ordering, and requires that we augment $\mathcal{T}$ with null values, and then allow a more complex ordering relation in which nulls represent "smaller" values than non-nulls. Although the null-augmented ordering is actually a generalization of the natural inclusion ordering, it is also substantially more complex to manage, and many of the results which are straightforward for the natural ordering case become quite complex in the null-augmented ordering case. To keep the presentation as understandable as possible, we therefore restrict our attention in this section and the next to the natural ordering, and then separately develop results for the null-augmented ordering in 2.3.

Formally, the *natural inclusion ordering* $\subseteq$ on $\mathsf{LDB}(\mathbf{D}, \mu)$ is via relation-by-relation inclusion. That is, $M_1 \subseteq M_2$ if and only if for all $R \in \mathsf{Rel}(\mathbf{D})$, $R^{M_1} \subseteq R^{M_2}$. The schema $\mathbf{D}$ is *compatible for* $\subseteq$ if it admits a least model under this ordering. In the case that $\mathbf{D}$ is compatible for $\subseteq$, the *underlying* $\perp$*-poset schema* is $\widetilde{\mathbf{D}}^{[\subseteq]} = (\mathsf{LDB}(\mathbf{D}, \mu), \subseteq, \perp)$, with $\subseteq$ as just defined and $\perp$ the least model.

2.1.7 Morphisms and Views We define morphisms logically rather than algebraically. As an intuitive guide, the reader may think of the definition of a morphism $\mathbf{D}_1 \to \mathbf{D}_2$ as a set of queries, one for each $R \in \mathsf{Rel}(\mathbf{D}_2)$, expressed in a relational calculus ([39, 10.2]) of the language of $\mathbf{D}_1$. The formal definition is as follows.

Let $\mathbf{D}_1$ and $\mathbf{D}_2$ be relational schemata. A *morphism* $f : \mathbf{D}_1 \to \mathbf{D}_2$ is an interpretation of $\mathsf{Lang}(\mathbf{D}_2)$ into $\mathsf{Lang}(\mathbf{D}_1)$ which is the identity on $\mathcal{T}$ and which is logically correct. More precisely, for each $R \in \mathsf{Rel}(\mathbf{D}_2)$ we are given a formula $\mathsf{Def}(f, R)$ in $\mathsf{Lang}(\mathbf{D}_1)$ (called the *interpretation formula* for R) with exactly the variables $\{v_1, .., v_{\mathsf{Ar}(R)}\}$ free. This in turn induces a function $f^* : \mathsf{DB}(\mathbf{D}_1, \mu) \to \mathsf{DB}(\mathbf{D}_2, \mu)$ which sends each $M \in \mathsf{DB}(\mathbf{D}_1, \mu)$ to the structure $f^*(M)$, with $R^{f^*(M)}$ the relation which is explicitly defined by the formula $\mathsf{Def}(f, R)$ relative to M. (See [36] for a detailed explanation of this idea, including many examples.) The variable v_i identifies the i^{th} column of this relation. (As a notational convention, we shall always use the variable v_i to mark the i^{th} column of an interpretation formula.) The interpretation f is *logically correct* if $f^*(\mathsf{LDB}(\mathbf{D}_1, \mu)) \subseteq \mathsf{LDB}(\mathbf{D}_2, \mu)$ for any type assignment μ. In other words, f is logically correct if it maps legal states of $\mathbf{D}_1$ into legal states of $\mathbf{D}_2$. In this case, the underlying function $f' : \mathsf{LDB}(\mathbf{D}_1, \mu) \to \mathsf{LDB}(\mathbf{D}_2, \mu)$ is defined to be the appropriate restriction of f^*. If $\mathbf{D}_1$ and $\mathbf{D}_2$ are compatible for $\subseteq$, then f is termed *compatible for* $\subseteq$ provided that f' defines a $\perp$-poset morphism in the sense of 1.1.3. This underlying $\perp$-poset morphism is denoted $\widetilde{f}^{[\subseteq]}$. As f' and $\widetilde{f}^{[\subseteq]'}$ denote the same function, we use the former notation in lieu of the latter. As in the order-theoretic case, the morphism $f : \mathbf{D}_1 \to \mathbf{D}_2$ is an *isomorphism* if there is another morphism $g : \mathbf{D}_2 \to \mathbf{D}_1$ such that both $f \circ g$ and $g \circ f$ are identities.

Two morphisms $f, g : \mathbf{D}_1 \to \mathbf{D}_2$ with the property that $f' = g'$ are termed *equivalent*. We do not distinguish between such morphisms, as they are just different logical representations for the same query. Also, morphisms may be composed in a natural sense. The reader is referred to [15] or [36] for details, and to the latter paper as well for a much more detailed presentation of the use of logical interpretations to define database mappings.

A *view* of the relational schema $\mathbf{D}$ is just a pair $\Gamma = (\mathbf{V}, \gamma)$ in which $\mathbf{V}$ is also a relational schema and $\gamma : \mathbf{D} \to \mathbf{V}$ is a morphism such that f' is surjective for *any* choice of μ. If $\mathbf{D}$, $\mathbf{V}$, and γ are each compatible for $\subseteq$, then Γ clearly defines a $\perp$-poset view in the sense of 1.1.8, which we denote by $\widetilde{\Gamma}^{[\subseteq]} = (\widetilde{\mathbf{D}}^{[\subseteq]}, \widetilde{\gamma}^{[\subseteq]})$. We call such a view a $\subseteq$-*view*. The $\subseteq$-view Γ is an *ideal* $\subseteq$-view if $\widetilde{\Gamma}^{[\subseteq]}$ is an ideal view of $\widetilde{\mathbf{D}}^{[\subseteq]}$, in the sense of 1.3.1.

2.1.8 Notation and Conventions For the rest of this section and throughout Section 2.2, unless specifically stated to the contrary (such as in examples), we assume that $\mathbf{D}$ is a relational schema which is compatible with $\subseteq$. Furthermore, when we say that $X = \{\Gamma_i \mid i \in I\}$ is a (direct) decomposition or is independent, we shall mean precisely that each Γ_i is a $\subseteq$-view, and that $\{\widetilde{\Gamma_i}^{[\subseteq]} \mid i \in I\}$ is a (direct) decomposition in the sense of 1.2.3.

2.1.9 The structure of ideal views Let $\Gamma = (\mathbf{V}, \gamma)$ be an ideal $\subseteq$-view of $\mathbf{D}$. Within the relational framework governed by the natural ordering, ideal views, as defined in the previous section, have a very particular structure. Namely, for any $R \in \mathsf{Rel}(\mathbf{D})$ and any $M \in \mathsf{LDB}(\mathbf{D}, \mu)$, we must have that $R^{\gamma'(M)} \subseteq R^M$, since $\epsilon\langle\widetilde{\mathbf{D}}^{[\subseteq]}, \widetilde{\mathbf{V}}^{[\subseteq]}\rangle'$ is quasi-contracting. This in turn implies that the interpretation formula $\mathsf{Def}(\gamma, R)$ must be expressible in the form $R(v_1, .., v_{\mathsf{Ar}(R)}) \wedge \Phi$, with Φ a formula in the language of $\mathbf{D}$ with at most $\{v_1, .., v_{\mathsf{Ar}(R)}\}$ free. The formula Φ is called the *restrictor* of R for Γ, and is denoted $\mathsf{Rstr}(\Gamma, R)$. This characterization immediately provides us with a representation for the meet of two ideal views, as given below.

2.1.10 Proposition *Let $\Gamma_1 = (\mathbf{V}_1, \gamma_1)$ and $\Gamma_2 = (\mathbf{V}_2, \gamma_2)$ be ideal $\subseteq$-views of $\mathbf{D}$. Then the view $\Gamma_3 = (\mathbf{V}_3, \gamma_3)$ defined by $\mathsf{Rstr}(\Gamma_3, R) = \mathsf{Rstr}(\Gamma_1, R) \wedge \mathsf{Rstr}(\Gamma_2, R)$ for each $R \in \mathsf{Rel}(\mathbf{D})$ is also an ideal $\subseteq$-view, and $\widetilde{\Gamma_3}^{[\subseteq]} = \widetilde{\Gamma_1}^{[\subseteq]} \cap \widetilde{\Gamma_2}^{[\subseteq]}$, in the sense of 1.3.7.* $\square$

One of the thornier issues in the theory of views of relational database schemata is that the axiomatization of a quite reasonably defined view may be far more complex than that of the base schema itself. For example, Hull [34, Lemma 4.1] provides an example of a single-relation schema constrained only by three functional dependencies, and a simple projective view of four of its five columns, such that the view schema is not finitely axiomatizable. In addition, it is not difficult to construct an equally simple example of a view consisting of two projections which is not axiomatizable by first-order means [31]. The following proposition, which not only states that finite-axiomatizability is preserved by ideal $\subseteq$-views, but which also provides an explicit formula for the axiomatization, is therefore quite important.

2.1.11 Proposition *Let $\Gamma = (\mathbf{V}, \gamma)$ be an ideal $\subseteq$-view of $\mathbf{D}$ and let Σ be a basis for $\mathsf{Con}(\mathbf{D})$. Then a basis for $\mathsf{Con}(\mathbf{V})$ is given by*

$$\Sigma \cup \{(\forall v_1.., v_{\mathsf{Ar}(R)})(R(v_1, .., v_{\mathsf{Ar}(R)}) \Rightarrow \mathsf{Rstr}(\gamma, R)) \mid R \in \mathsf{Rel}(\mathbf{D})\}.$$

In particular, if $\mathbf{D}$ is finitely axiomatizable, so too is $\mathbf{V}$.

PROOF: On the one hand, since γ' must be surjective, any $M \in \mathsf{LDB}(\mathbf{V}, \mu)$ must be the image under γ' of some $P \in \mathsf{LDB}(\mathbf{D}, \mu)$. Hence, any such M must satisfy the above constraints. On the other hand, if $M \in \mathsf{DB}(\mathbf{V}, \mu)$ satisfies the above constraints, then it is, *a fortiori*, in $\mathsf{LDB}(\mathbf{D}, \mu)$, and it furthermore must map to itself under γ'. Therefore, it is in $\mathsf{LDB}(\mathbf{V}, \mu)$. $\square$

Vardi [48, Thm. 3] has already established that a view which is a component in a direct decomposition of a finitely axiomatized schema must itself be finitely axiomatizable. The above proposition extends this result for order-compatible views, since an ideal view need not have a direct complement.

2.1.12 Tuple-based ideal views Let $\Gamma = (\mathbf{V}, \gamma)$ be an ideal $\subseteq$-view of $\mathbf{D}$. In general, the restrictor $\mathsf{Rstr}(\gamma, R)$ for a given $R \in \mathsf{Rel}(\mathbf{D})$ may be an arbitrarily complex first-order formula. It turns out, however, that under suitable circumstances we can establish that it has a particularly simple form, in which $\mathsf{Rstr}(\gamma, R)$ is totally independent of the current state of the database. The action of γ' then becomes one of simply examining each tuple of each relation, and, independently of any other tuples in the database, either discarding it or else retaining it for the view state. More formally, we say that Γ is *tuple-based* if for any $M_1, M_2 \in \mathsf{LDB}(\mathbf{D}, \mu)$, any $R \in \mathsf{Rel}(\mathbf{D})$, and any $\mathsf{Ar}(R)$-tuple $x \in R^{M_1} \cap R^{M_2}$, we have $x \in R^{\gamma'(M_1)}$ if and only if $x \in R^{\gamma'(M_2)}$.

2.1.13 Examples The notion of a tuple-based view is quite intuitive, and it is perhaps not immediately obvious how an ideal view can fail to have this property. Therefore, before we establish the conditions under which an ideal $\subseteq$-view must be tuple based, let us consider a few simple examples of $\subseteq$-views which are not tuple based, so that we have a better idea of what can go wrong.

Let the type algebra $\mathcal{T}$ have exactly three atomic types τ_A, τ_B, and τ_C, each admitting an infinite number of distinct domain values. Let $\mathbf{D}$ be the relational schema with four relational symbols $R_1[AB]$, $R_2[AB]$, $R_3[BC]$, and $R_4[ABC]$. Here we have used the domain names as abbreviations for constraints, as in 2.1.5. For example, $R_1[AB]$ means that R_1 is a binary relation symbol with the constraint $R_1(v_1, v_2) \Rightarrow (\tau_A(v_1) \wedge \tau_B(v_2))$. In addition to these domain constraints, assume that $\mathbf{D}$ is governed by the following constraint.

$$(\forall v_1, v_2, v_3)(R_4(v_1, v_2, v_3) \Leftrightarrow ((R_1(v_1, v_2) \vee R_2(v_1, v_2)) \wedge R_3(v_2, v_3)))$$

In other words, R_4 is the union of the two joins $R_1[AB] \bowtie R_3[BC]$ and $R_2[AB] \bowtie R_3[BC]$. Define the view $\Gamma_1 = (\mathbf{V}_1, \gamma_1)$ to be the ideal $\subseteq$-view which preserves R_1 identically, but drops the other three relations. Thus, $\mathsf{Rstr}(\gamma_1, R_1) = \mathbf{true}$,[4] but $\mathsf{Rstr}(\gamma_1, R_2) = \mathsf{Rstr}(\gamma_1, R_3) = \mathsf{Rstr}(\gamma_1, R_4) = \mathbf{false}$. Define $\Gamma_2 = (\mathbf{V}_2, \gamma_2)$ and $\Gamma_3 = (\mathbf{V}_3, \gamma_3)$ with respect to R_2 and R_3 similarly. Clearly each of these views is tuple based. Now $\Gamma_1 \vee \Gamma_3$ also exists as an ideal $\subseteq$-view $\Gamma_{13} = (\mathbf{V}_{13}, \gamma_{13})$. Indeed, put $\mathsf{Rstr}(\gamma_{13}, R_1) = \mathsf{Rstr}(\gamma_{13}, R_3) = \mathbf{true}$, $\mathsf{Rstr}(\gamma_{13}, R_2) = \mathbf{false}$, and $\mathsf{Rstr}(\gamma_{13}, R_4) = R_1(v_1, v_2) \wedge R_3(v_2, v_3)$. Then Γ_{13} is clearly this join, yet there is no way that we can make $\mathsf{Rstr}(\gamma_{13}, R_4)$ tuple based. We must select from R_4 only those tuples arising from the join of R_1 and R_3, and exclude those arising from the join of R_2 and R_3. Note also that $\{\Gamma_{13}, \Gamma_2\}$ is a direct complementary pair, so that this phenomenon applies even to views which are complemented.

The above example is somewhat anomalous in that it is not constrained by *data dependencies* in the sense of [19]. The constraint that R_4 is the union of two joins is not expressible as a data dependency. One might therefore still conjecture that if $\mathbf{D}$ has a basis of data dependencies,

[4] We use **true** (resp. **false**) to denote the sentence which is always true (resp. false).

then all ideal $\subseteq$-views will be tuple-based. The following example shows that conjecture to be false.

2.1.14 Example We modify the previous example slightly by adding two new atomics type τ_{W_1} and τ_{W_2}. Each of these types has exactly one possible domain value. The constant symbol a_1 represents the unique value for τ_{W_1}, and a_2 represents the unique value for τ_{W_2}. The definitions of R_1, R_2, and R_3 remain unchanged. We add relation $R_5[ABCW]$, defined as follows.

$$(\forall v_1, v_2, v_3, v_4)((R_5(v_1, v_2, v_3, v_4) \wedge \tau_{W_1}(v_4)) \Leftrightarrow (R_1(v_1, v_2) \wedge R_3(v_2, v_3)))$$
$$(\forall v_1, v_2, v_3, v_4)((R_5(v_1, v_2, v_3, v_4) \wedge \tau_{W_2}(v_4)) \Leftrightarrow (R_2(v_1, v_2) \wedge R_3(v_2, v_3)))$$

This new relation is just like R_4 of the previous example, except that we have used the extra column to tag the origin of the tuple. Now define the constraint for R_4 to be the following.

$$(\forall v_1, v_2, v_3)(R_4(v_1, v_2, v_3) \Leftrightarrow (\exists x)(R_5(v_1, v_2, v_3, x))$$

This will yield exactly the same R_4 as in the previous example. The join $\Gamma_{13} = \Gamma_1 \vee \Gamma_3$ will be the same as in the previous example, except that the restrictor $\mathsf{Rstr}(\gamma_{13}, R_5) = \mathbf{false}$ must be added to account for the new relation. Again note that $\{\Gamma_{13}, \Gamma_2\}$ is a direct complementary pair. This time, however, $\mathbf{D}$ has a basis consisting of data dependencies in the sense of [19]. Indeed, the only sentence specified above which is not a data dependency is the "$\Leftarrow$" direction of the constraint for R_4. But it may be replaced with the following pair of data dependencies.

$$(\forall v_1, v_2, v_3, v_4)((R_5(v_1, v_2, v_3, v_4) \wedge \tau_{W_1}(v_4)) \Rightarrow (R_1(v_1, v_2) \wedge R_3(v_2, v_3)))$$
$$(\forall v_1, v_2, v_3, v_4)((R_5(v_1, v_2, v_3, v_4) \wedge \tau_{W_2}(v_4)) \Rightarrow (R_2(v_1, v_2) \wedge R_3(v_2, v_3)))$$

The last example, while constrained by dependencies, is not constrained by *total* or *universal* dependencies. Rather, existential quantification is essential in the representation. On the other hand, the penultimate example, while constrained by universal sentences, is not constrained by data dependencies. If we combine the two conditions, and require universal dependencies, we get *total data dependencies*. Such dependencies are a special case of a more general class of first-order sentences, called *universal Horn sentences*, [43, 25.12], which imply an important model-theoretic property known as closure (or preservation) under intersections [41].

2.1.15 Closure under intersections The schema $\mathbf{D}$ is *closed under intersections* (or $\cap$-*closed*) if for any *nonempty* $Y \subseteq \mathsf{LDB}(\mathbf{D}, \mu)$, $\bigcap Y \in \mathsf{LDB}(\mathbf{D}, \mu)$ also.[5] Note that we do not require $\bigcap \emptyset$ to exist in $\mathsf{LDB}(\mathbf{D}, \mu)$. This would imply the existence of a largest model, which is unrealistic and unnecessary.

[5] In [41], the property is stated only for finite intersections, while we require intersections of arbitrary nonempty sets of models to be models. It is easy to see that such arbitrary intersections of models of a universal Horn theory is still a model; finiteness is not an issue. The reader should also be careful not to confuse closure under intersections with the *intersection property* [40, Def. 4.1], which is a much weaker property.

2.1.16 Lemma *Let* $\mathbf{D}$ *be* $\cap$*-closed, and let* $\Gamma = (\mathbf{V}, \gamma)$ *be an ideal* $\subseteq$*-view of* $\mathbf{D}$.

(a) *For any nonempty* $Y \subseteq \mathsf{LDB}(\mathbf{D}, \mu)$, *we have* $\gamma'(\bigcap Y) = \bigcap\{\gamma'(M) \mid M \in Y\}$.

(b) Γ *is tuple-based.*

PROOF: (a) Let $Y \subseteq \mathsf{LDB}(\mathbf{D}, \mu)$ be nonempty but otherwise arbitrary. We immediately have that $\gamma'(\bigcap Y) \subseteq \bigcap\{\gamma'(M) \mid M \in Y\}$. On the other hand, $\bigcap\{\gamma'(M) \mid M \in Y\} \subseteq \bigcap Y$, so $\gamma'(\bigcap\{\gamma'(M) \mid M \in Y\}) \subseteq \gamma'(\bigcap Y)$. But $\gamma'(\bigcap\{\gamma'(M) \mid M \in Y\}) = \bigcap\{\gamma'(M) \mid M \in Y\}$, since the latter is already in the ideal defined by Γ. Hence $\bigcap\{\gamma'(M) \mid M \in Y\} \subseteq \gamma'(\bigcap Y)$.
(b) Let $M_1, M_2 \in \mathsf{LDB}(\mathbf{D}, \mu)$, let $R \in \mathsf{Rel}(\mathbf{D})$, and let $x \in R^{M_1} \cap R^{M_2}$, with $x \in R^{\gamma'(M_1)}$ as well. Then $x \in R^{\gamma'(M_1) \cap M_2}$ as well. But $\gamma'(M_1) \cap M_2 \in \mathsf{LDB}(\mathbf{V}, \mu)$, and so $\gamma'(M_1) \cap M_2 = \gamma'(\gamma'(M_1) \cap M_2) = \gamma'(\gamma'(M_1)) \cap \gamma'(M_2) = \gamma'(M_1) \cap \gamma'(M_2)$. Hence $x \in R^{\gamma'(M_2)}$, so Γ is tuple based. □

2.1.17 Lemma *If* $\mathsf{Con}(\mathbf{D})$ *has a basis consisting of universal Horn sentences, then* $\mathbf{D}$ *is* $\cap$*-closed.*

PROOF: It is easy to verify directly that the set of models of any family of universal Horn sentences is closed under intersection of nonempty sets. The only point needing elaboration is the effect of the axioms in $\mathbf{A}$, which need not be universal Horn. However, we are requiring that all models be identical on $\mathcal{T}$, and so all intersections will be identical on this component. Therefore, the proof for universal Horn sentences works in this case as well. □

2.1.18 Theorem *Let* $\mathsf{Con}(\mathbf{D})$ *have a basis of universal Horn sentences. Then any decomposition of* $\mathbf{D}$ *(into ideal* $\subseteq$*-views) consists entirely of tuple-based views.*

PROOF: Combine 2.1.16 and 2.1.17. □

In particular then, if $\mathsf{Con}(\mathbf{D})$ has a basis consisting of total data dependencies, a decomposition into ideal $\subseteq$-views will always be tuple based. Since all $\subseteq$-order based decompositions may be so represented, this provides in some sense a theoretical justification for the intuitive notion of the simplicity and naturalness of universal Horn sentences in general and total data dependencies in particular.

It is possible to establish that the set of all tuple-based views of a given schema $\mathbf{D}$ is finite, and so will form a finite decomposition framework when $\mathbf{D}$ has the model intersection property. However, the proof is rather complex, and many of the resulting views are of limited interest in practice. Therefore, we shall not pursue that topic further in this paper. Rather, we now turn to a special but most interesting class of tuple-based views, those whose restrictors depend only upon the types of the elements in the tuple.

2.2 Restrictive Relational Decomposition

In this section, we continue to investigate decompositions with respect to the natural ordering $\subseteq$, but narrow the scope of our attention to those views whose restrictors look only at the types (in the sense of the type algebra $\mathcal{T}$) of the entries in the tuples. To motivate the importance of such views, we formulate the classical notion of a join-based decomposition within the ideal $\subseteq$-view framework.

2.2.1 Example Let us examine is some detail how to recapture the decomposition of a relation $R[ABCD]$ into the three projections $R[AB]$, $R[BC]$, and $R[CD]$. In the traditional approach [39, 7.7], all that we require is that the base schema $\mathbf{D}$ be constrained by the join dependency $\bowtie[AB, BC, CD]$, which is an abbreviation for the following sentence.

$$(\forall v_A, v_B, v_C, v_D, x_A, x_C, x_D, y_A, y_B, y_D)$$
$$((R(v_A, v_B, v_C, v_D) \wedge R(x_A, v_B, x_C, x_D) \wedge R(y_A, y_B, x_C, y_D)) \Rightarrow R(v_A, v_B, x_C, y_D)) \quad \text{(tjd)}$$

However, such a decomposition is only subdirect; the components are not independent. To achieve independence, we must allow null values [12]. In our formalization, we have five atomic types. The types τ_A, τ_B, τ_C, and τ_D are the atomic types corresponding to the domain names. Each of these four types is axiomatized to have infinitely many domain elements. The type τ_ν is the null type. It is axiomatized to have exactly one domain element, which is represented by the constant symbol ν. The schema $\mathbf{D}$ has the single relation symbol R. To the join dependency, we must add constraints identifying exactly where nulls may and may not appear. To do so, we introduce two parameterized abbreviations for sentences. For any $S \subseteq \{A, B, C, D\}$, let $\mathsf{ForbidNulls}(S)$ denote the following sentence.

$$(\forall v_A, v_B, v_C, v_D)((R(v_A, v_B, v_C, v_D) \wedge (\bigwedge_{i \in S} \tau_\nu(v_i))) \Rightarrow \mathbf{false}). \quad \text{(fn(S))}$$

In words, the sentence $\mathsf{ForbidNulls}(S)$ mandates that a tuple which has nulls in each of the positions identified by S cannot occur. We require that it hold for all minimal S which are not subsets of a complement (relative to $\{A, B, C, D\}$) of one of $\{A, B\}$, $\{B, C\}$, and $\{C, D\}$. In other words, we stipulate that $\mathsf{ForbidNulls}(S)$ hold for $S = \{A, C\}$, $S = \{B, C\}$, and $S = \{B, D\}$.

Similarly, for $S \in \{A, B, C, D\}$, let $\mathsf{RequireNulls}(S)$ denote the following sentence.

$$(\forall v_A, v_B, v_C, v_D)((R(v_A, v_B, v_C, v_D) \wedge (\bigwedge_{i \in S} \tau_i(v_i))) \Rightarrow R(\xi_A, \xi_B, \xi_C, \xi_D)). \quad \text{(rn(S))}$$

Here ξ_i is defined by

$$\xi_i = \begin{cases} v_i & \text{if } i \in S \\ \nu & \text{otherwise.} \end{cases}$$

The sentence $\mathsf{RequireNulls}(S)$ mandates that whenever a tuple t occurs with all positions identified by S nonnull, then the tuple obtained by replacing the entries in all other columns by nulls is also in that instance. We require that it hold for all S which are supersets of one of $\{A, B\}$, $\{B, C\}$, and $\{C, D\}$, except that we may omit $\{A, B, C, D\}$. In other words, we stipulate that $\mathsf{RequireNulls}(S)$ hold for all $S \in \{\{A, B\}, \{B, C\}, \{C, D\}, \{A, B, C\}, \{A, B, D\}, \{A, C, D\}, \{B, C, D\}\}$.

We cannot use the join dependency (tjd) directly, since it would indiscriminately include nulls in the joined tuples; rather, we must replace it with the following.

$$(\forall v_A, v_B, v_C, v_D, x_A, x_C, x_D, y_A, y_B, y_D)$$
$$((((\tau_A(v_A) \wedge \tau_B(v_B)) \vee (\tau_B(v_B) \wedge \tau_C(x_C)) \vee (\tau_C(x_C) \wedge \tau_D(y_D))) \wedge$$
$$R(v_A, v_B, v_C, v_D) \wedge R(x_A, v_B, x_C, x_D) \wedge R(y_A, y_B, x_C, y_D)) \Rightarrow R(v_A, v_B, x_C, y_D)) \quad \text{(j)}$$

Constraint (j) is the same as the traditional join dependency (tjd), except that it only selects those tuples with non-nulls in the right places. Indeed, in the absence of nulls (j) and (tjd) are identical, so that we could in fact have given (j) in place of (tjd), and we may regard (j) as "the" join dependency $\bowtie[AB, BC, CD]$, without sacrificing the classical case.

Define the ideal $\subseteq$-view $\Gamma_{AB} = (\mathbf{V}_{AB}, \gamma_{AB})$ to have $\mathsf{LDB}(\mathbf{V}_{AB}, \mu) = \{M \in \mathsf{LDB}(\mathbf{D}, \mu) \mid ((x_A, x_B, x_C, x_D) \in R^M) \Rightarrow ((x_C = \nu) \wedge (x_D = \nu))\}$. The view mapping γ_{AB} has $\mathsf{Def}(\gamma_{AB}, R) = (R(v_A, v_B, v_C, v_D) \wedge \tau_A(v_A) \wedge \tau_B(v_B) \wedge \tau_\nu(v_C) \wedge \tau_\nu(v_D))$. Note that Γ_{AB} is essentially the projection of the first two columns of R, because if (x_A, x_B, x_C, x_D) is any tuple in any instance of R, then (x_A, x_B, ν, ν) must be in that instance of R as well, since the constraint $\mathsf{RequireNulls}(\{A, B\})$ must hold. The two nulls provide no additional information, and serve only as placeholders. The views $\Gamma_{BC} = (\mathbf{V}_{BC}, \gamma_{BC})$ and $\Gamma_{CD} = (\mathbf{V}_{CD}, \gamma_{CD})$ are defined analogously with respect to the second and third, and third and fourth columns of R, respectively.

With these definitions, it is easy to see that $\{\Gamma_{AB}, \Gamma_{BC}, \Gamma_{CD}\}$ is a direct decomposition of $\mathbf{D}$ into ideal $\subseteq$-views, which very closely resembles the classical (subdirect) projective decomposition governed by the join dependency $\bowtie[AB, BC, CD]$. The differences are due to the need to explicitly include nulls in the formalism, in order that independence of the views may be realized.

It is important to note that all of the required constraints are universal Horn; in fact, they are total data dependencies. Therefore, axoimatization with nulls may be carried out without sacrificing this property.

The above example is a paradigm for generally representing join-based decompositions within our framework, and the reader should have no problem extending it to an arbitrary join dependency. Such decompositions are, however, only one way of decomposing a schema; horizontal decompositions in the spirit of Smith [46] are also possible. It is possible to develop the theory of joins in both the vertical and horizontal direction much further, but that would take us beyond the scope of this paper. Some results of that nature were announced in [29], and will be much more fully developed in a forthcoming paper.

We now turn to a formalization of the specific form of restrictor which underlies decompositions of the above form.

2.2.2 Type restrictions All of the views in the previous example are tuple based with the additional property that the restrictors depend only upon the types of the tuple entries. Such views are called type restrictions, and are sufficiently important to deserve special attention.

Let n be a nonnegative natural number. A *simple n-type* over $\mathcal{T}$ is an n-tuple of the form $(\tau_1, \tau_2, .., \tau_n)$, with each $\tau_i \in \mathbf{T} \setminus \{\tau_\perp\}$. This string is taken to be an abbreviation for the well-formed formula $\tau_1(v_1) \wedge .. \wedge \tau_n(v_n)$ with exactly $\{v_1, .., v_n\}$ free. A *compound n-type over $\mathcal{T}$* is a (possibly empty) set $S = \{s_1, s_2, .., s_k\}$ of simple n-types. This set S is an abbreviation for the disjunction of its elements; that is, $\bigvee_{i=1}^{k} s_i$. It is easy to see that any logical combination of simple n-types (but without quantifiers) is logically equivalent to a compound n-type. Indeed, let Ψ be such a logical combination. Without loss of generality, we may assume it to be in disjunctive normal form, that is, a disjunction of conjunctions of literals, with each literal either a simple n-type or its negation [15, Cor. 15C]. However, each negated simple n-type $\neg(\tau_1, .., \tau_n)$ is logically equivalent to the simple n-type $(\bar{\tau}_1, .., \bar{\tau}_n)$, and each conjunction $\bigwedge_{i=1}^{m}(\tau_{i1}, .., \tau_{in})$ may be replaced with $(\bigwedge_{i=1}^{m} \tau_{i1}, .., \bigwedge_{i=1}^{m} \tau_{in})$.

An ideal $\subseteq$-view $\Gamma = (\mathbf{V}, \gamma)$ with the property that $\mathsf{Rstr}(\gamma, R)$ is a compound $\mathsf{Ar}(R)$-type for each $R \in \mathsf{Rel}(\mathbf{D})$ is called a *type restriction*. Because we shall not consider other varieties of restrictions in this work, we shall henceforth refer to type restrictions as simply *restrictions*.

2.2.3 Examples of restrictions All of the views in 2.2.1 are restrictions. In 2.1.13, each of the views Γ_1, Γ_2, and Γ_3 is a restriction. The view Γ_{13} of 2.1.13 is a join of restrictions which is not itself a restriction, so that the class of restrictions is not closed under join. This is the case even for complemented views, as in 2.1.13 Γ_{13} has Γ_2 as its complement. In 2.1.14, exactly the same situation occurs. We do, however, have the following result.

2.2.4 Proposition *Let $\Gamma_1 = (\mathbf{V}_1, \gamma_1)$ and $\Gamma_2 = (\mathbf{V}_2, \gamma_2)$ be restrictions. Then the view $\Gamma_3 = (\mathbf{V}_3, \gamma_3)$ defined by $\mathsf{Rstr}(\gamma_3, R) = \mathsf{Rstr}(\gamma_1, R) \wedge \mathsf{Rstr}(\gamma_2, R)$ for each $R \in \mathsf{Rel}(\mathbf{D})$ is itself a restriction, and $\widetilde{\Gamma_3}^{[\subseteq]} = \widetilde{\Gamma_1}^{[\subseteq]} \cap \widetilde{\Gamma_2}^{[\subseteq]}$, in the sense of 1.3.7.*

PROOF: If $\mathsf{Rstr}(\gamma_1, R) = \{s_{11}.., s_{1m}\}$ and $\mathsf{Rstr}(\gamma_2, R) = \{s_{21}, .., s_{2n}\}$, then $\mathsf{Rstr}(\gamma_1, R) \wedge \mathsf{Rstr}(\gamma_2, R) = \{s_{1i} \wedge s_{2j} \mid 1 \leq i \leq m \text{ and } 1 \leq j \leq n\}$, which is a type restriction. Furthermore, if $M \in \mathsf{LDB}(\mathbf{V}_1, \mu) \cap \mathsf{LDB}(\mathbf{V}_2, \mu)$, then $(\gamma_1 \circ \gamma_2')(M) = M$. Thus Γ_3 as defined above is indeed a meet of Γ_1 and Γ_2. $\square$

We can overcome the problem of lack of closure under joins by enforcing some further conditions on the nature of $\mathsf{Con}(\mathbf{D})$. Recall from 2.1.16 that under the assumption of $\cap$-closure, all ideal views are tuple based. By enforcing a few additional properties, we can assure that these tuple-based views are in fact projections.

2.2.5 Model invariance conditions A constant symbol $c \in \mathbf{K}$ is *inessential* if there is a type $\tau_c \in \mathbf{T}$ such that $\mathbf{A} \models ((c = v_1) \Leftrightarrow \tau_c(v_1))$. In other words, τ_c is constrained to have only one element in any model, and c identifies that value. We say that $\mathcal{T}$ is *essentially constant free* if every $c \in \mathbf{K}$ is inessential. In other words, $\mathcal{T}$ cannot have any constant symbols other than those which are already identifiable as the unique element of an atomic type. The type algebras underlying the examples of 2.1.13, 2.1.14, and 2.2.1 are essentially constant free.

An *automorphism* α of $\mathcal{D}(\mu)$ is any bijection on that set which preserves type, in the precise sense that if $\tau \in \mathbf{T}$ and $x \in \mathcal{D}(\mu)$, then $x \in \mathsf{Dom}_\mu(\tau)$ if and only if $\alpha(x) \in \mathsf{Dom}_\mu(\tau)$. If α is such an automorphism and $t = (t_1, .., t_n)$ is an n-tuple of elements of $\mathcal{D}(\mu)$, $\alpha(t)$ denotes the n-tuple $(\alpha(t_1), .., \alpha(t_n))$. If $M \in \mathsf{LDB}(\mathbf{D}, \mu)$, $\alpha(M)$ denotes the structure obtained by simultaneously substituting $\alpha(x)$ for each occurrence of x in M, for each x in $\mathcal{D}(\mu)$. This includes the relations and constant symbols of $\mathcal{T}$. We also write $\alpha(\mu)$ to denote the type assignment obtained by the substitution. We say that $\mathcal{T}$ is *invariant under automorphisms* if $\alpha(\mu) = \mu$, and we say that $\mathbf{D}$ is *closed under automorphisms* if for any $M \in \mathsf{LDB}(\mathbf{D}, \mu)$ and any automorphism α of $\mathcal{D}(\mu)$, $\alpha(M) \in \mathsf{LDB}(\mathbf{D}, \mu)$ as well.

2.2.6 Lemma *If $\mathcal{T}$ is essentially constant free, then $\mathcal{T}$ is invariant under automorphisms and $\mathbf{D}$ is closed under automorphisms.*

PROOF: Note first that for any $M \in \mathsf{LDB}(\mathbf{D}, \mu)$, $\alpha(M) \in \mathsf{LDB}(\mathbf{D}, \alpha(\mu))$, since applying α just amounts to a renaming. But if $\mathcal{T}$ is essentially constant free, $\mu = \alpha(\mu)$. Hence $\mathsf{LDB}(\mathbf{D}, \mu) = \mathsf{LDB}(\mathbf{D}, \alpha(\mu))$, and so $\mathbf{D}$ is closed under automorphisms. $\square$

It is important to note that $\mathbf{D}$ is arbitrary in the above, and may actually be an ideal view of the schema to be decomposed.

2.2.7 Local column independence We say that $\mathbf{D}$ has *local column independence* if for any $M \in \mathsf{LDB}(\mathbf{D}, \mu)$ and any $R \in \mathsf{Rel}(\mathbf{D})$, if $s = (s_1, .., s_{\mathsf{Ar}(R)})$, $t = (t_1, .., t_{\mathsf{Ar}(R)}) \in R^M$ with $\mathsf{BaseType}(s_i, \mu) = \mathsf{BaseType}(t_i, \mu)$ for $1 \leq i \leq \mathsf{Ar}(R)$, then we have for any i and j that $s_i = s_j$ if and only if $t_i = t_j$.

In the traditional theory of dependencies of a single relation, a data dependency is call *typed* if the same variable cannot occur in more than one column of a given relation [19]. Clearly, if $\mathbf{D}$ has only one relation and is constrained by typed dependencies, then it will have local column independence. The schemata in each of 2.1.13, 2.1.14, and 2.2.1 have local column independence.

2.2.8 The least model property We say that $\mathbf{D}$ has the *least model property* if for any $R \in \mathsf{Rel}(\mathbf{D})$, any $M \in \mathsf{LDB}(\mathbf{D}, \mu)$, and any $t \in R^M$, there is a least database $M_t \in \mathsf{LDB}(\mathbf{D}, \mu)$ with $t \in R^{M_t}$. Observe that if $\mathbf{D}$ is $\cap$-closed, then it must have the least model property, since the intersection of all models containing t must then be M_t. In particular, if it is constrained by universal Horn sentences, it will have the least model property.

We are now in a position to establish the main decomposition theorem for restrictions.

2.2.9 Decomposition theorem for restrictions *Suppose that $\mathcal{T}$ is invariant under automorphisms, and $\mathbf{D}$ has local column independence and the least model property. Then every ideal $\subseteq$-view of $\mathbf{D}$ is a restriction, and the restrictions form a finite decomposition framework.*

PROOF: Let $R \in \mathsf{Rel}(\mathbf{D})$, let $M \in \mathsf{LDB}(\mathbf{D}, \mu)$, let $\Gamma = (\mathbf{V}, \gamma)$ be an ideal $\subseteq$-view of of $\mathbf{D}$, and let $s = (s_1, .., s_{\mathsf{Ar}(R)})$, $t = (t_1, .., t_{\mathsf{Ar}(R)}) \in R^M$ with $\mathsf{BaseType}(s_i, \mu) = \mathsf{BaseType}(t_i, \mu)$ for $1 \leq i \leq \mathsf{Ar}(R)$. Let α denote the automorphism which sends $s_i \mapsto t_i$ and $t_i \mapsto s_i$, $1 \leq i \leq \mathsf{Ar}(R)$, and leaves all other elements of $\mathcal{D}(\mu)$ fixed. Since $\mathbf{D}$ has local column independence, α is well defined. Let $M_s \in \mathsf{LDB}(\mathbf{D}, \mu)$ denote the least model with $s \in R^{M_s}$, and define M_t similarly. Assume that $t \in R^{\gamma'(M)}$; we will show that $s \in R^{\gamma'(M)}$ as well. Now $t = \alpha(s) \in R^{\alpha(M_s)}$, and so $M_t \subseteq \alpha(M_s)$. But α^2 is the identity, so $\alpha(M_t) \subseteq \alpha^2(M_s) = M_s$. But also $M_s \subseteq \alpha(M_t)$ since $s \in \alpha(M_t)$ and M_s is the least model containing s. Hence $\alpha(M_t) = M_s$. Since $\mathbf{V}$ is closed under automorphisms (by 2.2.6), we must have that $M_s \in \mathsf{LDB}(\mathbf{V}, \mu)$, whence $\gamma'(M_s) = M_s$, and so $s \in R^{\gamma'(M)}$.

The finiteness of the set of all restrictions follows immediately from the finiteness of $\mathbf{T}$, so in particular the set of all complemented restrictive views is finite. □

2.2.10 Corollary *Let $\mathbf{D}$ be a single-relation schema constrained by typed total data dependencies. Then $\mathbf{D}$ has a unique ultimate decomposition into a finite number of restrictions.* □

2.3 Projective Relational Decomposition

The formulation of the decomposition in 5.1 of $R[ABCD]$ into its AB, BC, and CD components may seem somewhat contrived, in that "subsumed" tuples must be present to make things work. For example, if the tuple (a, b, c, d) is present in the instance of R, then it really should not be necessary to include tuples such as (a, b, ν, ν), (ν, b, c, ν) or (ν, ν, c, d), since (a, b, c, d) in a sense carries more information. However, in the $\subseteq$-based order framework, these subsumed tuples are essential for the construction of the "projections" which are the components of the decomposition. To avoid this redundancy, we must build into the extant framework a formalism asserting that tuples such as (a, b, ν, ν) are subsumed by tuples such as (a, b, c, d), and then define a new order relation based upon such subsumption. In this section, we show how this may be done by augmenting the type algebra to include explicitly identified nulls. We then establish that this new augmented framework is isomorphic, in a strong sense, to a restrictive framework as developed in 2.2.

2.3.1 Augmentation of the type algebra Let $\mathcal{T} = (\mathbf{T}, \mathbf{K}, \mathbf{A})$ be a type algebra. The *associated null-augmented algebra* of $\mathcal{T}$, denoted $\mathsf{Aug}(\mathcal{T}) = (\mathsf{Aug}(\mathbf{T}), \mathsf{Aug}(\mathbf{K}), \mathsf{Aug}(\mathbf{A}))$, is defined as follows.

(a) To $\mathbf{K}$ we add one new constant symbol ν_τ for each $\tau \in \mathsf{Atoms}(\mathcal{T})$. ν_τ is called the *null constant of type* τ. Formally, $\mathsf{Aug}(\mathbf{K}) = \mathbf{K} \cup \mathbf{K}_\nu$, where $\mathbf{K}_\nu = \{\nu_\tau \mid \tau \in \mathbf{T} \setminus \{\tau_\perp\}\}$.

(b) To $\mathbf{T}$ we add one new atomic type ι_τ for each $\tau \in \mathsf{Atoms}(\mathcal{T})$. We call ι_τ the *atomic null type for* τ. The only value of type ι_τ is identified by the null constant ν_τ; thus $\mathsf{Aug}(\mathbf{A}) \models (\iota_\tau(x) \Leftrightarrow (x = \nu_\tau))$. These new types are disjoint from all existing types; *i.e.*, they are atoms in the null-augmented algebra. Of course, new nonatomic types will be constructed also; $\mathsf{Aug}(\mathbf{T})$ is the Boolean algebra of types generated by the atomic types in $\mathsf{Atoms}(\mathcal{T}) \cup \{\iota_\tau \mid \tau \in \mathsf{Atoms}(\mathcal{T})\}$.

(c) The set $\mathsf{Aug}(\mathbf{A})$ consists of the axioms of $\mathbf{A}$, together with the additional axioms necessary to make the conditions of (a) and (b) hold, including the necessary definitions of new nonatomic types.

For any type $\tau \in \mathsf{Atoms}(\mathcal{T})$, we define $\widehat{\tau} = \tau \vee \iota_\tau$. This definition is extended to each $\tau \in \mathbf{T}$ by setting $\widehat{\tau} = \bigvee\{\widehat{\omega} \mid \omega \in \mathsf{Atoms}(\mathcal{T}) \text{ and } \omega \leq \tau\}$. The type $\widehat{\tau}$ is called the *null completion* of τ. The set $\{\widehat{\tau} \mid \tau \in \mathbf{T} \setminus \{\tau_\perp\}\}$ is denoted by $\mathsf{NullAugTypes}(\mathcal{T})$. We also define $\check{\tau} = \bigvee\{\iota_\omega \mid \omega \in \mathsf{Atoms}(\mathcal{T}) \text{ and } \omega \leq \tau\}$. The type $\check{\tau}$ is called the *nullification* of τ. The set $\{\check{\tau} \mid \tau \in \mathbf{T} \setminus \{\tau_\perp\}\}$ is called the set of *null types*, and is denoted $\mathsf{NullTypes}(\mathcal{T})$. For consistency in defining formulas by rules, we admit $\check{\iota}_\tau$ as a synonym for ι_τ.

We continue to use the symbol $\tau_\top$ to denote the greatest type of $\mathcal{T}$; the symbol $\widehat{\tau_\top}$ will be used to denote the greatest type of $\mathsf{Aug}(\mathcal{T})$.

2.3.2 Convention From now on, $\mathbf{D}$ will denote a relational database schema taken over the type algebra $\mathsf{Aug}(\mathcal{T})$. We also assume henceforth that μ is a type assignment for $\mathsf{Aug}(\mathcal{T})$.

2.3.3 The semantics of nulls Nulls possess a very special semantics, which we use to define a new partial order $\preceq$ on $\mathsf{LDB}(\mathbf{D}, \mu)$. Informally, $\preceq$ starts with an ordering on domain

elements which states that the null ν_τ corresponding to given atomic type τ is less than any domain value of type τ, and extends this notion up through tuples to models. Formally, let $a = (a_1, a_2, .., a_n)$ and $b = (b_1, b_2, .., b_n)$ be n-tuples of elements from $\mathcal{D}(\mu)$. We say that b *subsumes* a, and write $a \preceq b$ just in case for each i, $1 \leq i \leq n$, exactly one of the following two conditions holds.

(i) $a_i = b_i$;

(ii) For some $\tau \in \mathsf{Atoms}(\mathcal{T})$, $a_i = \nu_\tau$ and $b_i \in \mathsf{Dom}_\mu(\widehat{\tau})$.

For n-ary relations $P, Q \subseteq \mathcal{D}(\mu)^n$, we write $P \preceq Q$ just in case for each $p \in P$ there is a $q \in Q$ with $p \preceq q$. For $M, N \in \mathsf{LDB}(\mathbf{D}, \mu)$, $M \preceq N$ denotes that $R^M \preceq R^N$ for each $R \in \mathsf{Rel}(\mathbf{D})$. We say that $\mathbf{D}$ is *compatible* for $\preceq$ if $\preceq$ is a partial order on $\mathsf{LDB}(\mathbf{D}, \mu)$ and if it admits a least model under this ordering. It is immediate that $\preceq$ is a preorder (*i.e.*, reflexive and transitive) on $\mathsf{LDB}(\mathbf{D}, \mu)$, but for it to be a partial order, we must have that $M \preceq N \preceq M$ implies $M = N$, which will be the case if and only if no relation in R^M contains any distinct tuples which are related under $\preceq$. In other words, no tuple is allowed in R^M if it is subsumed by another tuple in that relation.

While we give no specific semantics to our nulls beyond that implied by the underlying model theory, they are most closely related to the *no information* nulls of Zaniolo [50], and may be interpreted in that fashion if desired. Indeed, the ordering $\preceq$ then becomes subsumption in the sense of Zaniolo. Note that the one difference is that our nulls, by their very nature, give information about type, while those of Zaniolo do not. However, in a traditional typed domain framework, this makes no essential difference, since in that context the type of a null is completely determined by the column in which it appears.

Our interpretation does differ slightly from that of Chan and Mendelzon [12], in which the the nulls are "missing values", and are represented by the equivalent of unbound variables. With such an approach, it is somewhat more involved to say exactly what a decomposition is, since the decomposition and the base schema are not quite logically equivalent. While we feel that the issues surrounding incomplete information are important, we show here that they are not essential to a theory of decomposition. Nonetheless, the idea of using nulls in the base schema, and of projecting only tuples with enough "non-nullness", is central to both our approach and to that of Chan and Mendelzon.

More generally, certain interpretations of null values have proven an invaluable tool in the representation of special kinds of incomplete information databases. Among the major contributors to such research efforts have been Biskup [9], Imieliński and Lipski [35], and Levene and Loizou [38]. A comprehensive survey and investigation of these approaches and others may be found in the dissertation of Grahne [23]. It is extremely important to note that our use of null values does not imply the existence of any incomplete information. As pointed out in 1.1.7, in an incomplete information context, a database is not a single structure, but rather a set of structures. Tuples of the form (a, b, ν) are then used as *abbreviations* for an entire set of tuples, one for each admissible value of ν. In our approach, on the other hand, a database is a single structure, and ν is just another data value, identified with a special type class. While we do not preclude the existence of a theory of equivalence between our context and one of incomplete information, we make no claim of such.

The notions of *compatibility for* $\preceq$ for morphisms and views are defined in a manner completely analogous to that given for the natural ordering $\subseteq$ described in 2.1. Similarly, we adopt

the notation $\widetilde{\mathbf{D}}^{[\preceq]} = (\mathsf{LDB}(\widetilde{\mathbf{D}}^{[\preceq]}), \preceq, \bot)$, $\widetilde{\gamma}^{[\preceq]}$, and $\widetilde{\Gamma}^{[\preceq]} = (\widetilde{\mathbf{V}}^{[\preceq]}, \widetilde{\gamma}^{[\preceq]})$, for the underlying $\preceq$-order based concepts, in complete analogy with the definitions made in 2.1

2.3.4 Example Let us now recast the decomposition of 2.2.1 into this new framework. The schema $\mathbf{D}$ still has a single relational symbol R of arity four. But now, the type algebra $\mathcal{T}$ need have only four atomic types τ_A, τ_B, τ_C, and τ_D. The augmentation $\mathsf{Aug}(\mathcal{T})$ will have four additional atomic types ι_{τ_A}, ι_{τ_B}, ι_{τ_C}, and ι_{τ_D}, with corresponding null constants ν_{τ_A}, ν_{τ_B}, ν_{τ_C}, and ν_{τ_D}, respectively. The null positioning sentence $\mathsf{ForbidNulls}(-)$ of 2.2.1 requires a purely cosmetic adjustment to accommodate the typed nulls of this framework. For $S \subseteq \{A, B, C, D\}$, we redefine $\mathsf{ForbidNulls}(S)$ as follows.

$$(\forall v_A, v_B, v_C, v_D)((R(v_A, v_B, v_C, v_D) \wedge (\bigwedge_{i \in S} \iota_{\tau_i}(v_i))) \Rightarrow \mathbf{false}). \qquad \text{(fn(S)*)}$$

We mandate that $\mathsf{ForbidNulls}(S)$ hold for exactly the same subsets of $\{A, B, C, D\}$ as identified in 2.2.1, namely, $\{A, C\}$ and $\{B, D\}$. On the other hand, $\mathsf{RequireNulls}(-)$ is discarded entirely; we instead employ the new parameterized constraint $\mathsf{ReplaceNulls}(-)$; for $S \subseteq \{A, B, C, D\}$ we define $\mathsf{ReplaceNulls}(S)$ as follows.

$$\bigwedge_{j \notin S}((\forall v_A, v_B, v_C, v_D, x_A, x_B, x_C, x_D)((R(v_A, v_B, v_C, v_D) \wedge R(x_A, x_B, x_C, x_D) \\ \wedge (\bigwedge_{i \in S}(\tau_i(v_i) \wedge (v_i = x_i))) \wedge \tau_j(v_j)) \Rightarrow \tau_j(x_j))) \qquad \text{(rn(S))}$$

The constraint $\mathsf{ReplaceNulls}(S)$ states that if a tuple t occurs with nonnulls in each position identified by S, then it may not have a null in a given position $j \notin S$ if there is another tuple t' which matches t on all positions identified by S and which has a nonnull in position j. In other words, nulls can only occur if they are not subsumed. We require that $\mathsf{ReplaceNulls}(S)$ hold for exactly the sets $\{A, B\}$, $\{B, C\}$, and $\{C, D\}$.

Finally, we require that the join dependency (j) from 2.2.1 hold. It need not be changed in any way. It is the other axioms which state exactly where nulls can and cannot be, and not the main join dependency, which differentiates this "projective" model from its "restrictive" counterpart. It is immediate that $\preceq$ is compatible for this schema, as subsumed tuples are never present in the relation. Note also that, as was the case in 2.2.1, all of the constraints presented here are universal Horn.

Having developed this example, we now turn to the general representation of such views.

2.3.5 Projections For $R \in \mathsf{Rel}(\mathbf{D})$, an *extended simple R-type* over $\mathcal{T}$ is triple of the form $(R, (\omega_1, .., \omega_{\mathsf{Ar}(R)}), (\eta_1, .., \eta_{\mathsf{Ar}(R)}))$, in which each $\omega_i \in \mathsf{NullAugTypes}(\mathcal{T}) \cup \{\tau_\bot\}$, and each $\eta_i \in \{0, 1\}$. The semantics we wish to obtain are as follows. Given a database $M \in \mathsf{LDB}(\mathbf{D}, \mu)$ and R, we first compute the type restriction corresponding to $(\omega_1, .., \omega_{\mathsf{Ar}(R)})$. Then, for each column i with $\eta_i = 1$, we convert all entries in that column to the null constant corresponding to the base type of that entry; columns for which $\eta_i = 0$ are left intact. Note in particular that if each $\eta_i = 0$, then this reduces to the restriction $(\omega_1, .., \omega_{\mathsf{Ar}(R)})$. Note also that if any $\omega_i = \tau_\bot$, then no tuples are projected for the corresponding relation; this is how we achieve an empty projection.

As this restriction operation does not simply accept or reject tuples, but may additionally alter them by replacing some entries with the corresponding nulls, it is not possible to represent

the semantics formally as a restrictor formula, as we did for type restrictions. Formally, the string $(R, (\omega_1, .., \omega_{\mathsf{Ar}(R)}), (\eta_1, .., \eta_{\mathsf{Ar}(R)}))$ is taken to be an abbreviation for the well-formed formula $(\exists y_1, .., y_{\mathsf{Ar}(R)})(R(y_1, .., y_{\mathsf{Ar}(R)}) \wedge \lambda_1(v_1) \wedge .. \wedge \lambda_{\mathsf{Ar}(R)}(v_{\mathsf{Ar}(R)}))$ with exactly $\{v_1, .., v_{\mathsf{Ar}(R)}\}$ free, and with the $\lambda_i(v_i)$'s defined to be the following formulas.

$$\lambda_i(v_i) = \begin{cases} \omega_i(y_i) & \text{if } \eta_i = 0 \\ \omega_i(y_i) \wedge (\bigvee_{\tau \in \mathsf{Atoms}(\mathcal{T})} (\widehat{\tau}(y_i) \wedge \omega_i(\nu_\tau) \wedge (v_i = \nu_\tau))) & \text{if } \eta_i = 1. \end{cases}$$

The rather complicated looking formula for $\eta_i = 1$ computes the null constant corresponding to the type of y_i, provided that y_i is of type ω_i to begin with. Note that exactly one of the disjuncts of this formula will be true, since any domain value y_i has exactly one atomic type associated with it (its base type).

Given two extended simple R-types $s_1 = (R, (\omega_{11}, .., \omega_{1\mathsf{Ar}(R)}), (\eta_{11}, .., \eta_{1\mathsf{Ar}(R)}))$, $s_2 = (R, (\omega_{21}, .., \omega_{2\mathsf{Ar}(R)}), (\eta_{21}, .., \eta_{2\mathsf{Ar}(R)}))$, their *common action* $s_1 \sqcap s_2$ is the extended R-type $(R, (\omega_{11} \wedge \omega_{21}, .., \omega_{1\mathsf{Ar}(R)} \wedge \omega_{2\mathsf{Ar}(R)}), (\max\{\eta_{11}, \eta_{12}\}, .., \max\{\eta_{1\mathsf{Ar}(R)}, \eta_{2\mathsf{Ar}(R)}\})$.

An *extended compound* R-type is just a set $S = \{s_1, .., s_m\}$ of extended simple R-types. As in the case of type restrictions, S is taken to be an abbreviation for the disjunction of its elements. We extend the notion of common action to extended compound R-types by defining $S \sqcap T$ to be $\{s \sqcap t \mid s \in S \text{ and } t \in T\}$.

A $\preceq$-view $\Gamma = (\mathbf{V}, \gamma)$ is a view of $\mathbf{D}$ is defined analogously to a $\subseteq$-view; $\mathbf{V}$ and γ must be compatible with $\preceq$. For Γ to be an ideal $\preceq$-view, the ideals must be with respect to the ordering $\preceq$. An ideal $\preceq$-view $\Gamma = (\mathbf{V}, \gamma)$ with the property that $\mathsf{Def}(\gamma, R)$ is a compound extended $\mathsf{Ar}(R)$-type for each $R \in \mathsf{Rel}(\mathbf{D})$ is called a *projection*. In analogy to 2.2.4, we may characterize the meet of projective views as follows.

2.3.6 Proposition *Let $\Gamma_1 = (\mathbf{V}_1, \gamma_1)$ and $\Gamma_2 = (\mathbf{V}_2, \gamma_2)$ be projections. Then the view $\Gamma_3 = (\mathbf{V}_3, \gamma_3)$ defined by $\mathsf{Def}(\Gamma_3, R) = \mathsf{Def}(\Gamma_1, R) \sqcap \mathsf{Def}(\Gamma_2, R)$ for each $R \in \mathsf{Rel}(\mathbf{D})$ is itself a projection, and $\widetilde{\Gamma_3}^{[\preceq]} = \widetilde{\Gamma_1}^{[\preceq]} \cap \widetilde{\Gamma_2}^{[\preceq]}$, in the sense of 1.3.7.*

PROOF: The critical point is to note that, by definition, any projection which preserves an atomic type $\tau \in \mathsf{Atoms}(\mathcal{T})$ (in some column of some relation) must in fact preserve $\widehat{\tau}$. In other words, it must preserve the null corresponding to that atomic type as well. Observe also that the null gives full type information about the domain value from which it arose; each atomic type has its own distinct null. From this it is clear that the operations of restricting to a given type and conversion of all elements of that same type to the corresponding nulls commute with one another. The common action corresponds exactly to this computation, and mirrors the operation of conjunction of the case of type restrictions. The rest of the argument is exactly as in 2.2.4. □

In terms of closure under view joins in the general case, projections suffer from the same problems as restrictions. To directly formulate a result similar to 2.2.9, we would need to develop analogs of the notion of a tuple-based view and of the least model property within the context of the augmented type algebra. Such a development is quite complex. Fortunately, there is an alternative. We may represent $\preceq$-compatible schemata and views in an order-preserving way as $\subseteq$-compatible schemata and views, and then invoke the results of the previous section.

The cornerstone idea is to replace each $\preceq$-compatible schema (in which no relation in a model may contain a tuple which is subsumed by another in that same relation) with the corresponding one in which *all* subsumed tuples are present. The result is a $\subseteq$-compatible schema with exactly the same order properties.

2.3.7 Null-Completion The *null completion* of a relation $X \subseteq \mathcal{D}(\mu)^n$ is $\widehat{X} = \{x \in \mathcal{D}(\mu)^n \mid (\exists y \in X)(x \preceq y)\}$. The relation X is *null complete* if $X = \widehat{X}$. In other words, a relation is null complete if it is closed under subsumption of tuples. The state $M \in \mathsf{DB}(\mathbf{D}, \mu)$ is *null complete* if R^M is null complete for each $R \in \mathsf{Rel}(\mathbf{D})$. In the case of a $\preceq$-compatible schema, null completion converts $\preceq$ to $\subseteq$, as shown by the following lemma.

2.3.8 Order isomorphism lemma *Let* $\mathbf{D}$ *be a* $\preceq$*-compatible database schema, and let* $M, N \in \mathsf{DB}(\mathbf{D}, \mu)$*. Then* $M \preceq N$ *if and only if* $\widehat{M} \subseteq \widehat{N}$*.*

PROOF: Suppose that $M \preceq N$. Let $R \in \mathsf{Rel}(\mathbf{D})$, and let $x \in R^{\widehat{M}}$. Then there is some $y \in R^M$ such that $x \preceq y$. Since $M \preceq N$, there is some $z \in R^N$ such that $y \preceq z$. But then $x \in R^{\widehat{N}}$ also, since $R^{\widehat{N}}$ is closed under subsumption. Hence $\widehat{M} \subseteq \widehat{N}$.

Conversely, suppose that $\widehat{M} \subseteq \widehat{N}$. If $R \in \mathsf{Rel}(\mathbf{D})$ and $x \in R^M$, then $x \in R^{\widehat{N}}$ as well, which implies that there is a $y \in R^N$ such that $x \preceq y$. Hence $M \preceq N$, as was to be shown. □

This ordering isomorphism may be lifted to schemata, in the sense that we may convert any $\preceq$-compatible schema $\mathbf{D}$ to a $\subseteq$-compatible schema $\widehat{\mathbf{D}}$ which is isomorphic to $\mathbf{D}$ in both the logical and order-theoretic senses. The following lemma shows how to achieve this.

2.3.9 Schema isomorphism lemma *Let* $\mathbf{D}$ *be a* $\preceq$*-compatible schema. Then there is a* $\subseteq$*-compatible schema* $\widehat{\mathbf{D}}$ *with the following properties.*

(a) $\mathsf{LDB}(\widehat{\mathbf{D}}, \mu) = \{\widehat{M} \mid M \in \mathsf{LDB}(\mathbf{D}, \mu)\}$.

(b) *There is a database morphism* $\mathbf{i_D} : \mathbf{D} \to \widehat{\mathbf{D}}$ *with underlying function* $M \mapsto \widehat{M}$ *which is an isomorphism of relational schemata. Furthermore, the underlying function* $\mathbf{i_D}' : \mathsf{LDB}(\mathbf{D}, \mu) \to \mathsf{LDB}(\widehat{\mathbf{D}}, \mu)$ *defines a* $\perp$*-poset isomorphism* $(\mathsf{LDB}(\mathbf{D}, \mu), \preceq, \perp) \cong (\mathsf{LDB}(\widehat{\mathbf{D}}, \mu), \subseteq, \perp)$.

(c) *The schema* $\widehat{\mathbf{D}}$ *is finitely axiomatizable if and only if* $\mathbf{D}$ *is.*

PROOF: The idea of the proof is quite simple. By the previous lemma, $\widehat{X}$ and X mutually define one another. Thus, for any $M \in \mathsf{LDB}(\mathbf{D}, \mu)$, M and $\widehat{M}$ implicitly (at the level of functions) define one another. With the aid of Beth's theorem, we can lift these to explicit definitions (at the level of logical interpretations). We then use these explicit definitions to construct the appropriate view morphisms and axiomatizations. The details of all of this are, inescapably, somewhat technical. The reader unfamiliar with these aspects of logic may safely skip the rest of the proof.

We build a new schema $\mathbf{D}^\sharp$ with $\mathsf{Rel}(\mathbf{D}^\sharp) = \mathsf{Rel}(\mathbf{D}) \cup \{\widehat{R} \mid R \in \mathsf{Rel}(\mathbf{D})\}$. (Here $\widehat{R}$ is just a symbol, distinct from R, with $\mathsf{Ar}(\widehat{R}) = \mathsf{Ar}(R)$.) For each $R \in \mathsf{Rel}(\mathbf{D})$, define the formula ψ_R

with $v_1, .., v_{\mathsf{Ar}(R)}$ free as

$$(\exists x_1, .., x_{\mathsf{Ar}(R)})(R(x_1, .., x_{\mathsf{Ar}(R)}) \wedge (\bigwedge_{i=1}^{\mathsf{Ar}(R)} ((x_i = v_i) \vee \bigvee_{\tau \in \mathsf{Atoms}(\mathcal{T})} (\tau(x_i) \wedge \iota_\tau(v_i))))),$$

and put

$$\mathsf{Con}(\mathbf{D}^\sharp) = \mathsf{Con}(\mathbf{D}) \cup \{(\forall v_1, .., v_{\mathsf{Ar}(R)})(\widehat{R}(v_1, .., v_{\mathsf{Ar}(R)}) \Leftrightarrow \psi_R) \mid R \in \mathsf{Rel}(\mathbf{D})\}.$$

We then have that for any $M \in \mathsf{LDB}(\mathbf{D}^\sharp, \mu)$ and $R \in \mathsf{Rel}(\mathbf{D})$, the relation $\widehat{R}^M$ is the null completion of R^M. Now since $\mathbf{D}$ is $\preceq$-compatible, we must have that $\widehat{R}$ implicitly defines R for each $R \in \mathsf{Rel}(\mathbf{D})$, in the sense that in any $M_1, M_2 \in \mathsf{LDB}(\widehat{\mathbf{D}}, \mu)$, $\widehat{R}^{M_1} = \widehat{R}^{M_2}$ if and only if $R^{M_1} = R^{M_2}$. Thus $\widehat{R}$ implicitly defines R, and so by Beth's theorem [22, Thm. 6.6.2], it explicitly defines R via a formula with exactly $\{v_1, .., v_{\mathsf{Ar}(R)}\}$ free and whose only nonlogical symbols are R and those of $\mathcal{T}$. We denote this formula by θ_R.

Now we move the $\widehat{R}$'s to their own schema $\widehat{\mathbf{D}}$. Thus, $\mathsf{Rel}(\widehat{\mathbf{D}}) = \{\widehat{R} \mid R \in \mathsf{Rel}(\mathbf{D})\}$. Define the interpretation θ of $\mathsf{Lang}(\mathbf{D})$ into $\mathsf{Lang}(\widehat{\mathbf{D}})$ by letting $\mathsf{Def}(\theta, R) = \theta_R$ for each $R \in \mathsf{Rel}(\mathbf{D})$. Similarly, the interpretation ψ of $\mathsf{Lang}(\widehat{\mathbf{D}})$ into $\mathsf{Lang}(\mathbf{D})$ is defined by letting $\mathsf{Def}(\psi, R) = \psi_R$. By the above constructions in $\mathbf{D}^\sharp$, it is clear that θ and ψ are inverse to one another. We define $\mathsf{Con}(\widehat{\mathbf{D}})$ to be the set $\{\varphi^\theta \mid \varphi \in \mathsf{Con}(\mathbf{D})\}$. Here φ^θ denotes the interpretation of φ in θ, as defined in [36] or [15, p. 160]. It amounts to substituting the formula θ_R for each occurrence in φ of each $R \in \mathsf{Rel}(\mathbf{D})$, subject to the proper term substitution for the arguments of R. With this definition, we have mutually inverse isomorphisms $\mathbf{D} \xrightarrow{\psi} \widehat{\mathbf{D}} \xrightarrow{\theta} \mathbf{D}$. Thus, $\mathsf{LDB}(\widehat{\mathbf{D}}, \mu) = \{\widehat{M} \mid M \in \mathsf{LDB}(\mathbf{D}, \mu)\}$, as required by (a), and ψ provides $\mathbf{i_D}$ for (b). Note also that the constraints of $\mathsf{Con}(\mathbf{D})$ and $\mathsf{Con}(\widehat{\mathbf{D}})$ are in bijective correspondence under this definition. Particularly, one will have a finite basis if and only if the other does, thus establishing (c). Finally, the previous lemma guarantees that ψ will be an order isomorphism. □

2.3.10 Null completed schemata, morphisms, and views Let $\mathbf{D}$ be a $\preceq$-compatible database schema.

(a) The schema constructed in the previous lemma is called the *null completion* of $\mathbf{D}$, and is denoted by $\widehat{\mathbf{D}}$. The isomorphism $\mathbf{i_D} : \mathbf{D} \to \widehat{\mathbf{D}}$ is called the *lifting* of $\mathbf{D}$ to $\subseteq$, and is denoted by the symbol $\mathbf{i_D}$ henceforth.

(b) Given another $\preceq$-compatible schema $\mathbf{E}$ and a $\preceq$-compatible morphism $f : \mathbf{D} \to \mathbf{E}$, we define $\widehat{f} : \widehat{D} \to \widehat{E}$ to be $\mathbf{i_E} \circ f \circ {\mathbf{i_D}}^{-1}$.

(c) If $\Gamma = (\mathbf{V}, \gamma)$ is a $\preceq$-compatible view of $\mathbf{D}$, we define $\widehat{\Gamma} = (\widehat{\mathbf{V}}, \widehat{\gamma})$.

(d) For each extended simple R-type $s = (R, (\omega_1, .., \omega_{\mathsf{Ar}(R)}), (\eta_1, .., \eta_{\mathsf{Ar}(R)}))$, define the *associated* $\subseteq$*-restriction* $\widehat{\rho}(s)$ to be $(\sigma_1, .., \sigma_{\mathsf{Ar}(R)})$, where

$$\sigma_i = \begin{cases} \omega_i & \text{if } \eta_i = 0 \\ \breve{\omega}_i & \text{if } \eta_i = 1. \end{cases}$$

For a extended compound R-type S, define $\widehat{\rho}(S) = \bigvee\{\widehat{\rho}(s) \mid s \in S\}$.

2.3.11 View structure lemma *Let $\Gamma = (\mathbf{V}, \gamma)$ be a $\preceq$-compatible view of $\mathbf{D}$.*

(a) *$\widehat{\Gamma}$, as defined in 2.3.10(c) above, is a $\subseteq$-compatible view of $\widehat{\mathbf{D}}$. Furthermore, this association is injective; if Υ is another $\preceq$-compatible view on $\mathbf{D}$, then $\widehat{\Upsilon} = \widehat{\Gamma}$ implies $\Upsilon = \Gamma$.*

(b) *The assignment $\Gamma \mapsto \widehat{\Gamma}$ induces a bijective correspondence between the projections on $\mathbf{D}$ and the restrictions on $\widehat{\mathbf{D}}$, and is defined on a relation-by-relation basis by sending the compound extended R-type S to the restriction $\widehat{\rho}(S)$.*

PROOF: (a) is immediate, since $\widehat{\gamma} = \mathbf{i}_{\mathbf{V}} \circ \gamma \circ \mathbf{i}_{\mathbf{D}}^{-1}$, and by 2.3.9(b), $\mathbf{i}_{\mathbf{V}}$ and $\mathbf{i}_{\mathbf{D}}$ are isomorphisms in both the logical and order-theoretic sense.
(b) Suppose that $\Gamma = (\mathbf{V}, \gamma)$ is a projection on $\mathbf{D}$, let $R \in \mathsf{Rel}(\mathbf{D})$, and let $\mathsf{Def}(\gamma, R)$ be defined by the compound extended R-type S. It is immediate that $\widehat{\rho}(S)$ gives $\mathsf{Def}(\widehat{\gamma}, R)$, so that $\widehat{\gamma}$ is indeed a restriction. To prove the converse, the key is to observe that if $\Gamma_1 = (\mathbf{V}_1, \gamma_1)$ is any restriction of $\widehat{\mathbf{D}}$, then for each $R \in \mathsf{Rel}(\mathbf{D})$, $\mathsf{Def}(\gamma_1, R)$ has a representation as a compound extended R-type S with each $s \in S$ of the form $(\xi_1, .., \xi_{\mathsf{Ar}(R)})$, with each $\xi_i \in \mathsf{NullTypes}(\mathcal{T}) \cup \mathsf{NullAugTypes}(\mathcal{T})$. This is because the schemata $\widehat{\mathbf{D}}$ and $\widehat{\mathbf{V}_1}$ are null complete, and so if any type $\tau \in \mathbf{T}$ is preserved in any column, its associated null types must be as well. But such a representation is precisely the image of a projection, as defined in 2.3.10(d) above. Hence the association between projections and restrictions is a bijective one, as required. □

2.3.12 Decomposition lifting theorem *Assume that $\mathbf{D}$ is an $\preceq$-compatible schema, and let X be a finite set of $\preceq$-compatible views of $\mathbf{D}$. Let $Y = \{\widehat{\Gamma_i} \mid \Gamma_i \in X\}$. Then $\{\widetilde{\Gamma}^{[\preceq]} \mid \Gamma \in X\}$ is a direct decomposition of $\widetilde{\mathbf{D}}^{[\preceq]}$ if and only if $\{\widetilde{\widehat{\Gamma}}^{[\subseteq]} \mid \Gamma \in Y\}$ is a direct decomposition of $\widehat{\mathbf{D}}$. Furthermore, the set X corresponds to a projective decomposition if and only if Y corresponds to a restrictive decomposition, with the projection-restriction identification given by 2.3.11(b).*

PROOF: To show that Y is a decomposition whenever X is, it suffices to note that the association $\Gamma \mapsto \widehat{\Gamma}$ preserves the order-theoretic properties of the views, in view of 2.3.9(b) and 2.3.11(a). The projective-restrictive association follows immediately from 2.3.11(b). □

We are finally in a position to assert the projection equivalent of 2.2.9.

2.3.13 Decomposition theorem for projections *Assume that $\mathbf{D}$ is $\preceq$-compatible, that $\mathcal{T}$ is closed under automorphisms, and that $\widehat{\mathbf{D}}$ has local column independence and the least model property. Then every ideal $\preceq$-view of $\mathbf{D}$ is a projection, and the projections form a finite decomposition framework for $\mathbf{D}$.*

PROOF: Combine 2.2.9 with the above theorem. □

The major drawback to the above corollary is that some of the properties must be checked in $\widehat{\mathbf{D}}$, and not in $\mathbf{D}$. It is easy to see that $\widehat{\mathbf{D}}$ will have local column independence if and only if $\mathbf{D}$ does, so this check may be moved to $\mathbf{D}$. However, the least model property does not translate so readily. Indeed, even if $\mathbf{D}$ has the least model property, its least models may not correspond to those of $\widehat{\mathbf{D}}$. To work directly within $\mathbf{D}$, we would have to formulate a more general notion of model intersection, based upon $\preceq$ rather than $\subseteq$. Such a formulation is far from simple.

The conclusion seems to be that, from a mathematical point of view, one is best off working in the $\subseteq$-compatible framework. The $\preceq$-compatible framework should be regarded as a "tuple conserving" implementation trick. From a theoretical point of view, our strong isomorphism theorems show this to be no compromise at all.

2.4 Horizontal Relational Decomposition

The decomposition of a relation schema into a set of its projections is generally known in the literature as *vertical decomposition* [44, Chap. 4], for obvious reasons. *Horizontal decomposition*, on the other hand, refers to splitting a relation into two or more relations, each of which is a subset of the original [44, Chap. 5]. Now the characterization that all components in a direct decomposition must be (up to isomorphism) ideal views (1.3.10) casts serious doubt that such a vertical/horizontal distinction can be made on a formal level. This is particularly reinforced by the example of 2.2.1, in which we show that "vertical" decomposition into independent components is in fact totally a horizontal restriction, when suitably viewed. Indeed, the use of this more general framework opens up an entirely new perspective on relational decomposition, which includes but is not limited to traditional vertical and horizontal decompositions. A complete presentation of such decompositions is beyond the scope of this paper; preliminary results were given in [29]. Nonetheless, it is important here to provide a minimal discussion of the degree to which what is known as horizontal decomposition is recaptured in our framework. We provide two illustrative examples.

2.4.1 Example In [46], Smith proposed decompositions based upon types of attributes. While this form of decomposition has subsequently received little attention in the literature, we feel that it is nonetheless an important one. Let us illustrate the general idea with a simple example, couched within our formalism.

Consider the schema $\mathbf{D}$ consisting of a single binary relation symbol R with the attribute assignment $R[EP]$. The domain E represents employees, and the domain P represents projects, with $R(x, y)$ meaning that employee x works on project y. Suppose that projects are further broken down into two categories, classified and unclassified. The constraint is that an employee may work on any number of unclassified projects, but may work on at most one classified project. This defines a horizontally embedded functional dependency, in the sense of [46]. What this implies is that the attribute P is really the "disjoint union" of the attribute N, denoting unclassified projects, and C, denoting classified projects. To represent this within our framework, each of these attributes has a type in the underlying type algebra. Denote these types as τ_E, τ_P, τ_N, τ_C. The types τ_E, τ_N, and τ_C are atomic with $\tau_N \vee \tau_C = \tau_P$. The main typing constraint for the relation R is

$$(\forall x)(\forall y)(R(x, y) \Rightarrow \tau_E(x) \wedge \tau_P(y)).$$

The horizontally embedded functional dependency is given by

$$(\forall x)(\forall y)(\forall z)(R(x, y) \wedge R(x, z) \wedge \tau_C(y) \wedge \tau_C(z) \Rightarrow (y = z))).$$

As a logical extension of the standard notation for ordinary functional dependencies, we may denote this horizontally embedded functional dependency by $E \rightarrow P[\tau_C]$.

Let $\Gamma_N = (\mathbf{V}_N, \gamma_N)$ be the ideal view with $\mathbf{V}_N$ defined to have the additional constraint (to those of $\mathbf{D}$)

$$(\forall x)(\forall y)(R(x,y) \Rightarrow \tau_E(x) \wedge \tau_N(y)),$$

which restricts the second column of R to τ_N. Define the database mapping $\gamma_N : \mathbf{D} \to \mathbf{V}_N$ by the formula $\mathsf{Def}(\gamma_N, S) = (R(v_1, v_2) \wedge \tau_N(v_2))$, which just says that R in $\mathbf{V}_N$ is the restriction of R in $\mathbf{D}$ to unclassified projects. That is, $\mathsf{Rstr}(\Gamma_N, R) = \tau_N(v_2)$. Similarly, let $\Gamma_C = (\mathbf{V}_C, \gamma_C)$ denote the ideal view with $\mathbf{V}_C$ defined to have the following additional constraints (to $\mathbf{D}$).

$$(\forall x)(\forall y)(R(x,y) \Rightarrow \tau_E(x) \wedge \tau_C(y))$$

$$(\forall x)(\forall y)(\forall z)(R(x,y) \wedge R(x,z) \Rightarrow (y = z))$$

The first says that R has the attribute constraint $R[EC]$, while the second says that R obeys the functional dependency $E \to C$. Define the database mapping $\gamma_C : \mathbf{D} \to \mathbf{V}_C$ by the formula $\mathsf{Def}(\gamma_C, R) = R(v_1, v_2) \wedge \tau_C(v_2))$, which says that R in $\mathbf{V}_C$ is the restriction of R in $\mathbf{D}$ to classified projects. It is not difficult to see that $\{\Gamma_N, \Gamma_C\}$ forms a complementary pair of views of $\mathbf{D}$, and defines a decomposition quite different from the traditional projective variety.

2.4.2 Example In [14, 44], a form of horizontal decomposition based upon *exceptions* is forwarded. The basic idea may be illustrated within the context of a simple example. Suppose the schema $\mathbf{D}$ has a single binary relation symbol $R[AB]$. There are no formal dependencies other than type constraints. However, the functional dependency $A \to B$ holds most of the time, but there are exceptions. The desired decomposition separates those tuples for which $A \to B$ holds from those for which it does not. Specifically, we decompose $\mathbf{D}$ into two views $\Gamma_{A \to B} = (\mathbf{V}_{A \to B}, \gamma_{A \to B})$ and $\Gamma_{A \not\to B} = (\mathbf{V}_{A \not\to B}, \gamma_{A \not\to B})$, each also with the single relation symbol $R[AB]$. In the view $\Gamma_{A \to B}$, we project just those tuples (a, b) with the property that if (a, b_1) is another tuple in the instance of R in $\mathbf{D}$, then $b = b_1$. In other words, this view projects those tuples which have no conflicts regarding the functional dependency $A \to B$. All other tuples are projected into $\Gamma_{A \not\to B}$. In the terminology of [14], the schema of the latter view satisfies the *afunctional* dependency $A \not\to B$. The pair $\{\Gamma_{A \to B}, \Gamma_{A \not\to B}\}$ clearly forms a subdirect complementary pair, as their union of their databases provides the database of $\mathbf{D}$. However, such a decomposition can never be direct; the two components are interdependent by their very nature, and so such decompositions are not recaptured by our theory.

3. Conclusions and Further Directions

We have shown that, under very reasonable conditions which are met by many practical examples, direct complements of database schemata are unique. This means that "incompatible" decompositions of a database schema cannot arise, and that any two direct decompositions have a common refinement. The theory is general; rather than being restricted to a small part of the relational approach, it is potentially applicable to a very wide variety of database models.

We have also provided selected insights into the application of this theory to the relational model. Most importantly, we have shown that the theory completely recaptures the notion of decomposition of a unirelational schema into projections, with reconstruction governed by a join dependency, reinforcing clearly the critical rôle of null values in this context. When the

relational schema is further constrained by universal Horn dependencies, we have shown that the only possible views are of a very special type which may be computed in time linear in the size of the database.

Our presentation on the application of this model has only scratched the surface, and there are many further directions which should now be taken. We briefly identify the most important ones which we are currently pursuing.

A theory of subdirect decompositions and their applications All of the results of this paper are for direct decompositions, in which the components are independent. For the purpose of decomposing a database schema into components which can be independently maintained and updated, direct decomposition is the natural form to require. However, subdirect decompositions, in which the components are dependent upon one another, play a key rôle in other database contexts. We are currently developing results in two of these areas.

In addressing the view update problem under the constant-complement strategy [5], demanding direct complements is needlessly restrictive. In [30], we have laid the groundwork for expanding upon the pioneering work of Bancilhon and Spyratos [5], by providing conditions for unique subdirect complements and further structural constraints which appear desirable for the constant-complement update strategy. There, rather than moving to a $\perp$-poset context (which does not appear to provide the tools necessary to achieve the desired uniqueness results), we have focused more closely on properties of general relational models. Specifically, to achieve a uniqueness results for subdirect complements, we introduce a sort of logical (rather than set-theoretic) monotonicity.

Another context in which subdirect rather than direct complements arise is that of the classical theory of acyclicity of relational schemata [18], [7]. In [32], we have provided a generalization of these classical results. The context of this generalization is a framework even more general than that presented in this paper. Working within the most general context introduced by Bancilhon and Spyratos ([4], [5]) in which database schemata are just sets and views are just surjective functions suffices to recover most of the key results. The key is to focus upon properties of the congruences generated by the views.

A theory of decomposition for incomplete-information databases The application of our results to data models other than the traditional relational one is also an important direction. We feel that the most important candidate at this time is the general incomplete information relational model, identified in 1.1.7. Although relatively little of a practical nature has been developed for this framework, its mathematical underpinnings are nonetheless very well understood, being just first-order model theory. Perhaps the major problem plaguing this framework is its extreme generality; it places so few restrictions on what can be represented that most problems, such as the view update problem, are completely intractable [28] [49]. We believe that by understanding what kinds of conditions must be imposed for such a schema to be decomposed, we may begin to see what kind of restrictions should be made to move closer to computational tractability of updates.

Acknowledgements

The initial part of this work was performed during the summer of 1989 while the author was visiting the CERFIM research center in Locarno, Switzerland, and it was completed while

visiting the Department of Mathematics at the University of Oslo, Norway. He is grateful for the hospitable environment and support provided by both of these institutions. He also wishes to thank the directorate of CERFIM for their willingness to include the preliminary version of this paper in their technical report series.

An anonymous referee gave this paper a most thorough reading, and provided numerous invaluable suggestions for improvement of the presentation.

References

1. Aho, A. V., Beeri, C., and Ullman, J. D., The theory of joins in relational databases, *ACM Trans. Database Systems*, **4**(1979), 297–314.

2. Alagić, S., "Relational Database Technology," Springer-Verlag, 1986.

3. Alagić, S., "Object-Oriented Database Programming," Springer-Verlag, 1989.

4. Bancilhon, F. and Spyratos, N., Independent components of databases, in: "Proceedings of the Seventh International Conference on Very Large Data Bases," 398–408, 1981.

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

6. Beeri, C., Bernstein, P. A., and Goodman, N., A sophisticate's introduction to database normalization theory, in: "Proceedings of the Fourth International Conference on Very Large Data Bases," 113–124, 1978.

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

8. Birkhoff, G., "Lattice Theory, third edition," American Mathematical Society, 1967.

9. Biskup, J., A formal approach to null values in database relations, in: Gallaire, H., Minker, J., and Nicolas, J. M., eds., "Advances in Data Base Theory," Volume 1, 299–341, Plenum Press, 1981.

10. Ceri, S., Gottlob, G., and Tanca, L., What you always wanted to know about Datalog (and never dared to ask), *IEEE Trans. Knowledge Data Engrg.*, **1**(1989), 146–166.

11. Ceri, S., Gottlob, G., and Tanca, L., "Logic Programming and Databases," Springer-Verlag, 1990.

12. Chan, E. P. F. and Mendelzon, A. O., Independent and separable database schemes, *SIAM J. Computing*, **16**(1987), 841–851.

13. Codd, E. F., Further normalization of the data base relational model, in: Rustin, R., ed., "Data Base Systems," 33–64, Prentice-Hall, 1972.

14. De Bra, P. and Paredaens, J., Horizontal decompositions for handling exceptions to functional dependencies, in: Gallaire, H., Minker, J., and Nicolas, J. M., eds., "Advances in Data Base Theory," Volume 2, 123–141, Plenum Press, 1984.

15. Enderton, H. B., "A Mathematical Introduction to Logic," Academic Press, 1972.

16. Fagin, R., Normal forms and relational database operators, in: "Proceedings of the ACM-SIGMOD 1979 International Conference on Management of Data," 153–160, 1979.

17. Fagin, R., A normal form for relational databases that is based on domains and keys, *ACM Trans. Database Systems*, **6**(1981), 387–415.

18. 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.

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

20. Gallaire, H. and Minker, J., eds., "Logic and Data Bases," Plenum Press, 1978.

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

22. Gallier, J. H., "Logic for Computer Science," John Wiley and Sons, 1986.

23. Grahne, G., The problem of incomplete information in relational databases, Technical Report A-1989-1, University of Helsinki, 1989, Also Ph.D. dissertation.

24. Grätzer, G., "Universal Algebra," D. Van Nostrand, 1968.

25. Grätzer, G., "General Lattice Theory," Academic Press, 1978.

26. Hegner, S. J., Algebraic aspects of relational database decomposition, in: "Proceedings of the Second ACM SIGACT-SIGMOD Symposium on Principles of Database Systems," 400–413, 1983.

27. Hegner, S. J., Canonical view update support through Boolean algebras of components, in: "Proceedings of the Third ACM SIGACT-SIGMOD Symposium on Principles of Database Systems," 163–172, 1984.

28. Hegner, S. J., Specification and implementation of programs for updating incomplete information databases, in: "Proceedings of the Sixth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems," 146–158, 1987.

29. 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," 174–183, 1988.

30. 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," 422–436, Springer-Verlag, 1990.

31. Hegner, S. J., Some open problems on view axiomatization, *Bulletin of the EATCS*, **40**(1990), 496–498.

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

33. Herrlich, H. and Strecker, G. E., "Category Theory," Allyn and Bacon, 1973.

34. Hull, R., Finitely specifiable implicational dependency families, *J. Assoc. Comp. Mach.*, **31**(1984), 210–226.

35. Imieliński, T. and Lipski, Jr., W., Incomplete information in relational databases, *J. Assoc. Comp. Mach.*, **31**(1984), 761–791.

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

37. Keller, A. M. and Ullman, J. D., On complementary and independent mappings on databases, in: "Proceedings of the 1984 Annual ACM-SIGMOD Conference," 143–148, 1984.

38. Levene, M. and Loizou, G., A domain-theoretic approach to incomplete information in nested relational databases, in: Litwin, W. and Schek, H., eds., "Foundations of Data Organization and Algorithms, 3rd International Conference, FODO 1989, Paris, France, June 1989, Proceedings," 439–456, Springer-Verlag, 1989.

39. Maier, D., "The Theory of Relational Databases," Computer Science Press, 1983.

40. Makowsky, J. A., Why Horn formulas matter in computer science: initial structures and generic examples, *J. Comput. System Sci.*, **34**(1987), 266–292.

41. Makowsky, J. A. and Vardi, M. Y., On the expressive power of data dependencies, *Acta Inform.*, **23**(1986), 231–244.

42. Minker, J., "Foundations of Deductive Databases and Logic Programming," Morgan Kaufmann, 1987.

43. Monk, J. D., "Mathematical Logic," Springer-Verlag, 1976.

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

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

46. Smith, J. M., A normal form for abstract syntax, in: "Proceedings of the Fourth International Conference on Very Large Data Bases," 156–162, 1978.

47. Ullman, J. D., "Principles of Database and Knowledge-Base Systems, Volume I," Computer Science Press, 1988.

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

49. Winslett, M. S., Updating databases with incomplete information, Technical Report STAN-CS-87-1143, Stanford University, 1987, Also Ph.D. dissertation.

50. Zaniolo, C., Database relations with null values, *J. Comput. System Sci.*, **28**(1984), 142–166.