Annals of Mathematics and Artificial Intelligence (2006) 46: 38–97
DOI: 10.1007/s10472-005-9013-y

The complexity of embedded axiomatization for a class of closed database views*

Stephen J. Hegner

Umeå University, Department of Computing Science, SE-901 87 Umeå, Sweden
E-mail: hegner@cs.umu.se

It is well known that the complexity of testing the correctness of an arbitrary update to a database view can be far greater than the complexity of testing a corresponding update to the main schema. However, views are generally managed according to some protocol which limits the admissible updates to a subset of all possible changes. The question thus arises as to whether there is a more tractable relationship between these two complexities in the presence of such a protocol. In this paper, this question is addressed for closed update strategies, which are based upon the constant-complement approach of Bancilhon and Spyratos. The approach is to address a more general question – that of characterizing the complexity of axiomatization of views, relative to the complexity of axiomatization of the main schema. For schemata constrained by *denial* or *consistency constraints*, that is, statements which rule out certain situations, such as the equality-generating dependencies (EGDs) or, more specifically, the functional dependencies (FDs) of the relational model, a broad and comprehensive result is obtained in a very general framework which is not tied to the relational model in any way. It states that every such schema is governed by an equivalent set of constraints which embed into the component views, and which are no more complex than the original set. For schemata constrained by *generating dependencies*, of which tuple-generating dependencies (TGDs) in general and, more specifically, both join dependencies (JDs) and inclusion dependencies (INDs) are examples within the relational model, a similar result is obtained, but only within a context known as meet-uniform decompositions, which fails to recapture some important situations. To address the all-important case of relational schemata constrained by both FDs and INDs, a hybrid approach is also developed, in which the general theory regarding denial constraints is blended with a focused analysis of a special but very practical subset of the INDs known as *fanout-free unary inclusion dependencies* (fanout-free UINDs), to obtain results parallel to the above-mentioned cases: every such schema is governed by an equivalent set of constraints which embed into the component views, and which are no more complex than the original set. In all cases, the question of view update complexity is then answered via a corollary to this main result.

Keywords: database, complexity, view

* Parts of this paper are based upon work reported in [21].

1. Introduction

In a seminal work [3], Bancilhon and Spyratos showed how well-behaved update strategies for database views can be modelled in a very general framework using the so-called constant complement strategy. Despite the classical nature of this work, it has seen significant recent application; for example, in [13], these ideas are applied to problems in the synchronization of tree-structured data.

The theory of constant-complement update strategies has also been advanced in recent years. In [19] and [20], it is shown that by augmenting this basic framework with natural order structure, true uniqueness for so-called order-based updates may be obtained, in the sense that there is but one way to represent an update to the view in terms of an update to the main schema, regardless of the choice of complement.

In addition to the questions of which updates to a view to allow, and how to reflect those updates back to the main schema, the question of tractability is also of fundamental importance. Specifically, it is important to know how difficult it is to determine whether a proposed view update is admissible under the constraints inherited from the main schema. Issues surrounding this complexity question are the focus of study in this paper.

Example 1.1 (A simple view with complex constraints). The complexity of the constraints which govern a view can be much greater than those of the schema over which the view is taken, as the following simple example illustrates. Let $\mathbf{E}_1$ be the single-relation schema with relation name $R[ABCD]$ on four attributes, constrained by the set $\mathcal{F}_1 = \{A \to D, B \to D, CD \to A\}$ of functional dependencies (FDs). For a relation M not to satisfy the constraints of $\mathcal{F}_1$, it must fail to satisfy at least one of the FDs, and this may be determined by identifying a specific pair of tuples in M for which that FD fails. In other words, to check whether M satisfies the constraints of $\mathcal{F}_1$, it suffices to test each pair of tuples in M. Thus, such testing has complexity bound $O(n^2)$ for (worst-case) time, with n the size of the relation.[1]

Now consider the simple projection view Π_{ABC} on $\mathbf{E}_1$, which maps any relation M on $R[ABCD]$ to its projection $\pi_{ABC}(M)$ on $R[ABC]$. As shown in Appendix A, this view is not finitely axiomatizable. More precisely, for any odd positive integer k, there is a ternary relation M_k, containing exactly k tuples, with the property that there is no relation N on $R[ABCD]$ which satisfies the dependencies in $\mathcal{F}_1$ and for which $\pi_{ABC}(N) = M_k$, yet for every proper subset $M' \subsetneq M_k$, there is a relation N' on $R[ABCD]$ which satisfies the dependencies in $\mathcal{F}_1$ and for which $\pi_{ABC}(N') = M'$. In other words, it is not possible to test, in time $O(n^k)$ for any finite k, whether a given ternary relation

[1] It is of course possible to do better in certain cases via the use of appropriate data structures. Indeed, upon using the number of disk accesses as the measure of complexity, key constraints may be checked in constant time [28]. However, the focus of this paper is a general theory based only upon the size of databases; the nuances of such special cases will not be considered.

M is of the form $\pi_{ABC}(N)$ for some relation N on $R[ABCD]$ which satisfies the constraints of $\mathcal{F}_1$.

While this example shows that checking the legality of a candidate state of a view is not a universally easy one, it does not establish that it is universally difficult either. Testing an arbitrary proposed update to a view for correctness is far more general a task than is testing a proposed update under a disciplined update strategy. A simple example, based upon example 1.1, makes this clear.

Example 1.2 (The complexity of view update). Let $\mathbf{E}_2$ be the relational schema with the five-attribute relation $S[ABCDE]$, constrained by $\mathcal{F}_2 = \mathcal{F}_1 \cup \{A \to E\}$. The pair of projections $\{\Pi_{ABCD}, \Pi_{ABCE}\}$, with the obvious semantics, forms a lossless decomposition, since the FD $A \to E$ implies $ABC \to E$, which implies the join dependency $\bowtie [ABCE, ABCD]$ [26, Thm. 3.7]. Thus, Π_{ABCD} is a complement of Π_{ABCE} in the classical sense of [3]. The decomposition into these two views is furthermore dependency preserving, since $\mathcal{F}_1$ embeds in Π_{ABCD}, while $\{A \to E\}$ embeds in Π_{ABCE}. Now consider updating the view Π_{ABCE} while holding Π_{ABCD} constant. All constraints in $\mathcal{F}_1$ will be satisfied after the update, since these constraints embed in Π_{ABCD}. Thus, to guarantee that a the new relation M on $S[ABCE]$ is legal, it suffices to check that it satisfies the single constraint $A \to E$. This observation is critical because Π_{ABCE} is not finitely axiomatizable, for the same reason that $\mathbf{E}_1$ of example 1.1 is not. The knowledge that the complement is held constant while performing updates is critical in keeping the complexity within bounds.

The above example notwithstanding, it is reasonable to ask why it is desirable to characterize the constraints within the views themselves. It is quite possible simply to take a proposed update to the view, reflect is back into the main schema using the constant-complement strategy, and accept it iff the resulting state of the main schema satisfies all of the constraints. The complexity of testing an update in this fashion depends only upon properties of the constraints on the main schema, and not upon those of the views. For example, if the main schema is constrained by FDs (as is $\mathbf{E}_2$ in example 1.2), then the complexity of testing admissibility of a view update is $O(n^2)$, with n the number of tuples in the main schema. (Note that a join must be computed to check update correctness in this way, but the time complexity to construct it is also $O(n^2)$.)

The goal of the work reported here, as well as that of the preceding work [15, 19, 20], is to investigate view update strategies which are *closed*. Roughly speaking, this means that updating a view should be no different than updating a main schema. The constraints on the view, as well as the updates which are allowed, must be understandable within the context of the view alone. In particular, it must not be necessary to consult the complementary view (or, equivalently, the main schema) to determine whether a view update is admissible. Clearly, the strategy of reflecting the view update back to the main schema, and performing the test for correctness there, is not closed.

While not all situations involving view updates demand closed strategies, it is certainly reasonable to expect that many will. The ideas surrounding this topic, with many illustrating examples, are developed in detail in [20, Section 1], There it is argued that closed update strategies have further desirable properties; more precisely, they are precisely those which are free of update anomalies.

A pair of views which supports a closed update strategy is called a *meet complementary pair*. In this case, there is a view which is common to these two, called the *meet*, and it is necessary and sufficient to hold this meet constant to ensure that updates respect the constant-complement strategy. The details are rather technical; they are developed in great detail in [20, Section 2], and summarized in 2.1 of this paper. It is, however, possible to give a rather simple characterization within a common relational context. Specifically, if $R[\mathbf{U}]$ is a relational schema constrained by a set of full dependencies, and $\mathbf{W}_1, \mathbf{W}_2 \subseteq \mathbf{U}$, then $\{\Pi_{\mathbf{W}_1}, \Pi_{\mathbf{W}_2}\}$, the pair of views defined by the projections of $R[\mathbf{U}]$ onto these attributes, forms a meet-complementary pair iff the associated decomposition is both lossless and dependency preserving, in the classical sense [20, 2.17]. Furthermore, the meet in this case is just $\Pi_{\mathbf{W}_1 \cap \mathbf{W}_2}$, the projection onto the common column of the two views. An example will help illustrate.

Example 1.3 (Closed and non-closed update strategies). Continue with the example $\mathbf{E}_2$ of example 1.2. The set $\{\Pi_{ABCD}, \Pi_{ABCE}\}$ forms a meet complementary pair, since the decomposition is both lossless and dependency preserving. The meet of these views is $\Pi_{ABCD \cap ABCE} = \Pi_{ABC}$. The theory of closed updates ensures that to check whether an update to Π_{ABCE} under constant complement Π_{ABCD} satisfies the constraints, it suffices to check whether it holds Π_{ABC} constant (in addition to verifying that it satisfies the local view constraints – in this case $\{A \to E\}$). Note that Π_{ABC} may be regarded as a view of $S[ABCE]$; thus, whether the update is admissible does depends only upon the state of $S[ABCE]$, and not upon the specific state of $S[ABCD]$. In other words, the test for admissibility of an update to Π_{ABCE} may be checked entirely within that view, provided that it is known that the current state of Π_{ABCD} is legal.

Now, let $\mathbf{E}_3$ have the same five-attribute relation $S[ABCDE]$, but with the additional constraint $D \to E$. That is, the constraint set is $\mathcal{F}_3 = \mathcal{F}_2 \cup \{D \to E\}$. This decomposition is not dependency preserving (although it is lossless), since $D \to E$ cannot be embedded in either view. Furthermore, the associated update strategy on Π_{ABCE} with constant complement Π_{ABCD} is not closed. Indeed, let $M_1 = \{(a_1, b_1, c_1, d_1, e_1), (a_2, b_2, c_2, d_2, e_1)\}$, and let $M_2 = \{(a_1, b_1, c_1, d_1, e_1), (a_2, b_2, c_2, d_1, e_1)\}$, Clearly, each is a legal state of $\mathbf{E}_3$. In each case, the associated state of Π_{ABCE} is $N = \{(a_1, b_1, c_1, e_1), (a_2, b_2, c_2, e_1)\}$. Consider the update to N which changes (a_1, b_1, c_1, e_1) to (a_1, b_1, c_1, e_2). This update is legal if the state of $\mathbf{E}_3$ is M_1, but not if it is M_2. Thus, the admissibility of this proposed update depends upon not only the state of Π_{ABCE}, but also on the state of the complement Π_{ABCD}. Hence, the associated strategy is not closed.

The theory developed in this paper includes the decomposition of $\mathbf{E}_2$ into Π_{ABCD} and Π_{ABCE}, but excludes the same decomposition of $\mathbf{E}_3$.

These examples illustrate that the question of the complexity of checking the correctness of candidate view updates with respect to a closed update strategy may be answered by studying a more general problem – that of determining the complexity of the constraints in an embedded cover (into the views of the decomposition) for the constraints of the main schema. It is primarily this latter question which forms the topic of investigation in this paper. From these examples, it might appear that the answer to this question is a rather trivial one, since the constraints which embed into the views are precisely those which are specified on the main schema. However, this presupposes that the constraints on the main schema are already expressed in a fashion which allows them to be embedded into the views. With FDs, this is not a problem, since it is easy to determine when an FD embeds into a projection (although in general it is necessary to determine whether some *cover* of the FDs embed [26, Def. 3.7]), but with more general families of constraints, particularly those which generate new elements, the representation of the constraints on the main schema as an equivalent family of constraints which embed in the views is far from trivial. Furthermore, there is no guarantee that this re-representation of full constraints as embedded ones can be achieved with no change of complexity.

The main results of this paper are along two dimensions. For schemata constrained by rules which check consistency (*e.g.*, the equality generating dependencies (EGDs) [26, Section 3.6] of the classical relational theory), a result is established, in a very general set-based context, independently of any particular data model, which states that there is an embedded cover of those constraints into the views which is no more complex than the original constraints. On the other hand, for situations in which rules which generate new elements are allowed, (*e.g.*, the tuple-generating dependencies (TGDs) of the classical relational theory, including the implicational dependencies and inclusion dependencies of the classical relational theory [26, Section 3.6]), the general results are somewhat more limited. It does not appear that the techniques for the consistency-checking rules can be extended in a way that includes the behavior of such dependencies within a broad range of applications.

To remedy this limitation, at least partly, an investigation of decompositions of multi-relational schemata constrained by EGDs together with fanout-free unary inclusion dependencies (UINDs) is also conducted. The approach combines the general results for constraint-checking dependencies with a detailed investigation of how such UINDs decompose. The result is a positive one – both the EGDs and the UINDs have embedded covers within the views with no increase in complexity. It is also a potentially practical one, since UINDs are adequate to model referential integrity constraints, or foreign-key dependencies, which are commonly employed in real-world database systems. While these results, which are found in Section 6, are formally based upon the the framework presented earlier, the reader may nonetheless wish to look through this section to obtain an idea of the power of this approach within the relational setting.

The organization of this paper is as follows. Section 2 provides a concise overview of the key ideas of the constant-complement update strategy, including in particular recent work on uniqueness within order-based frameworks. In Section 3, the principles of *PF-schemata* and *PF-views*, the abstraction of these database concepts used in this paper, is elaborated and related to those used in earlier papers. In Section 4, the key ideas of the decomposition results of the earlier paper [20] are translated to this framework, and specialized notions essential to the measure of complexity are developed. In Section 5, the general decomposition results, including in particular the embedding theorems for consistency-checking constraints, are established. Section 6 contains the detailed investigation within the context of multi-relational schemata constrained by both EGDs and UINDs. Section 7 offers some further directions and comparisons to the literature. Finally, Appendix A provides the details surrounding the axiomatizability issues of the example scheme $\mathbf{E}_1$ of example 1.1.

To close this section, some basic information on posets and relational notation which will be used throughout the paper is presented.

Discussion 1.4 (Posets, isomorphisms, and basic relational notation). A partially ordered set (*poset*) is a pair $\mathbf{P} = (P, \leqslant)$ in which P is a set and $\leqslant$ is a partial order on P. For $\mathbf{P}_i = (P_i, \leqslant)$ posets for $i = 1$ and $i = 2$, a morphism $f : \mathbf{P}_1 \to \mathbf{P}_2$ is a function $f : P_1 \to P_2$ with the property that for all $x, y \in P$, $x \leqslant y$ implies $f(x) \leqslant f(y)$. For any (not necessarily finite) set X, the poset $\boldsymbol{\mathscr{P}}_f(X) = (\mathscr{P}_f(X), \subseteq)$ consists of all finite subsets of X, ordered by set inclusion. For additional background on posets, the reader is referred to [9].

As a general principle, following the standard mathematical convention [2, 3.8], in all contexts an *isomorphism* will be taken to be a morphism which has both an left and a right inverse. Thus, if $f : X \to Y$ is an morphism in some context $\mathcal{C}$, it is an isomorphism iff there is a morphism $g : X_1 \to X_2$ with the properties that $g \circ f = \mathbf{1}_{X_1}$ and $f \circ g = \mathbf{1}_{X_2}$, with $\mathbf{1}_{X_i}$ the identity morphism on X_i. In the context of sets and functions, an isomorphism is a bijection, while in the context of posets, it is a bijection which both preserves and reflects order; *i.e.*, f is a poset isomorphism iff it is a bijection with the additional property that, for all $x, y \in P$, $f(x) \leqslant f(y)$ iff $x \leqslant y$.

Frequently, examples will be based upon the classical relational model, and the terminology and notation which has become standard over the past 30 years will be used without additional clarification. For any questions, the reader is referred to [26] or [1]. For this paper, it suffices to remark that given a universal relational schema $R[\mathbf{U}]$ on attribute set $\mathbf{U}$, and a subset $\mathbf{W} \subseteq \mathbf{U}$, $\Pi_{\mathbf{W}}$ will be used to denote the projective view of $R[\mathbf{U}]$ onto the attributes in $\mathbf{W}$, while $\pi_{\mathbf{W}} : R[\mathbf{U}] \to R[\mathbf{W}]$ will be used to denote the database mapping underlying this view. The notation $\mathsf{Dom}(A)$ will be used to denote the *domain* of A; *i.e.*, the set of values associated with the attribute $A \in \mathbf{U}$.

Finally, the notation $\mathsf{Card}(A)$ will be used to denote the cardinality of the set A.

2. An overview of previous work

While the results presented in this article do not deal with update strategies explicitly, they do address complexity issues which are cast within a context in which view updates are modelled and analyzed. Therefore, it is helpful to understand the underlying constant-complement update strategy. To provide the reader with this essential background, this section contains two summaries. Summary 2.1 recaps the necessary aspects of closed update strategies within the relevant set-based framework. It provides ideas which originated with the work of Bancilhon and Spyratos of [3], although it is recast within the formalism of [19, 20]. Summary 2.2 sketches the key ideas developed in [19, 20] which are necessary to extend the set-based ideas to the order-based context. These summaries include only the notions which are essential to the understanding of the current paper; the reader is referred to the aforementioned references for details and further clarification. Since [20] is an expanded and corrected version of [19], for the most part, only references to [20] will be given, even in the case that both papers contain the relevant material.

Summary 2.1 (The classical results in the set-based framework). In the original work of Bancilhon and Spyratos [3], a database schema $\mathbf{D}$ is just a set. To maintain consistency with the more structured frameworks to be introduced shortly, this set will be denoted $\mathsf{LDB}(\mathbf{D})$ and called the *legal databases* of $\mathbf{D}$. Thus, a database schema is modelled by its instances alone; constraints, schema structure, and the like are not represented explicitly. A *morphism* $f : \mathbf{D}_1 \to \mathbf{D}_2$ of database schemata is a function $f : \mathsf{LDB}(\mathbf{D}_1) \to \mathsf{LDB}(\mathbf{D}_2)$. A *view* of the schema $\mathbf{D}$ is a pair $\Gamma = (\mathbf{V}, \gamma)$ in which $\mathbf{V}$ is a schema and $\gamma : \mathbf{D} \to \mathbf{V}$ is a surjective database morphism. A *morphism* $f : \Gamma_1 \to \Gamma_2$ of views $\Gamma_1 = (\mathbf{V}_1, \gamma_1)$ and $\Gamma_2 = (\mathbf{V}_2, \gamma_2)$ is a morphism $f : \mathbf{V}_1 \to \mathbf{V}_2$ of schemata such that the diagram of figure 1 commutes. The *congruence* of Γ is the equivalence relation on $\mathsf{LDB}(\mathbf{D})$ defined by $(M_1, M_2) \in \mathsf{Congr}(\Gamma)$ iff $\gamma(M_1) = \gamma(M_2)$. It is easy to see that the views $\Gamma_1 = (\mathbf{V}_1, \gamma_1)$ and $\Gamma_2 = (\mathbf{V}_2, \gamma_2)$ of $\mathbf{D}$ are isomorphic iff $\mathsf{Congr}(\Gamma_1) = \mathsf{Congr}(\Gamma_2)$.

A pair $\{\Gamma_1 = (\mathbf{V}_1, \gamma_1), \Gamma_2 = (\mathbf{V}_2, \gamma_2)\}$ of views of the schema $\mathbf{D}$ is called a *subdirect complementary pair* if it defines a lossless decomposition of $\mathbf{D}$. More precisely, the product $\Gamma_1 \times \Gamma_2 = (\mathbf{V}_1 {}_{\gamma_1}\!\otimes_{\gamma_2} \mathbf{V}_2, \gamma_1 \otimes \gamma_2)$ has $\mathsf{LDB}(\mathbf{V}_1 {}_{\gamma_1}\!\otimes_{\gamma_2} \mathbf{V}_2) = \{(\gamma_1(M), \gamma_2(M)) \mid M \in \mathsf{LDB}(\mathbf{D})\}$. The *decomposition morphism* $\gamma_1 \otimes \gamma_2 : \mathbf{D} \to \mathbf{V}_1 {}_{\gamma_1}\!\otimes_{\gamma_2} \mathbf{V}_2$ is given on elements by $M \mapsto (\gamma_1(M), \gamma_2(M))$. The set $\{\Gamma_1, \Gamma_2\}$ of views forms a

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

Figure 1. View morphism.

subdirect complementary pair, and Γ_1 and Γ_2 are called *subdirect complements* of one another, just in case $\gamma_1 \otimes \gamma_2$ is a bijection. In other words, $\{\Gamma_1, \Gamma_2\}$ is a subdirect complementary pair precisely in the case that the state of the schema $\mathbf{D}$ is recoverable from the combined states of $\mathbf{V}_1$ and $\mathbf{V}_2$.

To define an update strategy on a view, the way in which such updates are reflected back to the main schema must be specified. In a *constant complement update strategy*, as first presented in [3], a (subdirect) complement is held constant. Thus, to update Γ_1 from N_1 to N_2 in the above context, the state of the schema $\mathbf{V}_2$ of the complement Γ_2 is held fixed. Since the decomposition morphism is injective, it is easy to see that there can be only one way to translate an update in such a context.

A fundamental condition in the study of view updates, as reported in [19, 20], and [21], is that the update strategies should be *closed*. The formal definition of an update strategy, and the list of conditions which render it closed, are quite technical and are not needed to understand the work reported here. The reader is therefore referred to [20, 3.1] for the complete, formal specification. For this work, it suffices to understand the basic idea that a *closed update strategy* has the property that whether or not a given update is allowed does not depend upon the particular state of the complement. Thus, if a given update from $N_1 \in \mathsf{LDB}(\mathbf{V}_1)$ to $N_1' \in \mathsf{LDB}(\mathbf{V}_1)$ is allowed for some state $N_2 \in \mathsf{LDB}(\mathbf{V}_2)$ of the complement, then it must be allowed for all states $N_2' \in \mathsf{LDB}(\mathbf{V}_2)$ with the property that $(N_1, N_2') \in \mathsf{LDB}(\mathbf{V}_{1\gamma_1}\!\otimes_{\gamma_2}\mathbf{V}_2)$.

Although every closed update strategy is definable via constant complement, not every subdirect complement of a view Γ_1 gives rise to closed update strategy. The property that a constant-complement update strategy be closed was termed Γ_2-translatability in [3, Def. 5.1]. However, it was not characterized explicitly until [15, 2.10], in which it was shown that Γ_2-translatability is characterized precisely by the property that the congruences of the views commute. Formally, the set $\{\Gamma_1, \Gamma_2\}$ of views of $\mathbf{D}$ is called a *fully commuting pair* if $\mathsf{Congr}(\Gamma_1) \circ \mathsf{Congr}(\Gamma_2) = \mathsf{Congr}(\Gamma_2) \circ \mathsf{Congr}(\Gamma_1)$, with '$\circ$' denoting ordinary composition of binary relations; i.e., $(M_1, M_2) \in \mathsf{Congr}(\Gamma_1) \circ \mathsf{Congr}(\Gamma_2)$ iff there is an $M_3 \in \mathsf{LDB}(\mathbf{D})$ such that $(M_1, M_3) \in \mathsf{Congr}(\Gamma_1)$ and $(M_3, M_2) \in \mathsf{Congr}(\Gamma_2)$. A subdirect complementary pair $\{\Gamma_1, \Gamma_2\}$ which is fully commuting is called a *meet-complementary pair*, and Γ_1 and Γ_2 are called *meet complements* of one another. In this case, $\mathsf{Congr}(\Gamma_1) \circ \mathsf{Congr}(\Gamma_2)$ is also an equivalence relation on $\mathsf{LDB}(\mathbf{D})$, and so it is possible to define (up to isomorphism) the view $\Gamma_1 \wedge \Gamma_2 = (\mathbf{V}_{1\gamma_1}\!\wedge_{\gamma_2}\mathbf{V}_2, \gamma_1 \wedge \gamma_2)$ with $\mathsf{Congr}(\Gamma_1 \wedge \Gamma_2) = \mathsf{Congr}(\Gamma_1) \circ \mathsf{Congr}(\Gamma_2)$, called the *meet* of $\{\Gamma_1, \Gamma_2\}$. It is, in effect, a well-behaved greatest lower bound of $\{\Gamma_1, \Gamma_2\}$. The situation is summed up in figure 2. Note that $\mathbf{V}_{1\gamma_1}\!\wedge_{\gamma_2}\mathbf{V}_2$ is (the schema of) a view not only of $\mathbf{D}$, but of $\mathbf{V}_1$ and $\mathbf{V}_2$ as well. The mappings $\lambda\langle\Gamma_i, \Gamma_1 \wedge \Gamma_2\rangle : \mathbf{V}_i \to \mathbf{V}_{1\gamma_1}\!\wedge_{\gamma_2}\mathbf{V}_2$ for $i = 1, 2$ are the morphisms of the associate *relative views* (see [20, 2.8]; or definition 3.12 for a development within the special context of this paper). In the context of updates, if $\{\Gamma_1, \Gamma_2\}$ forms a meet-complementary pair, then an update to the schema of Γ_1 is allowed with constant complement Γ_2 iff that update keeps the state of the meet schema $\mathbf{V}_{1\gamma_1}\!\wedge_{\gamma_2}\mathbf{V}_2$ constant. Since the meet schema may be

$$\begin{tikzcd}
 & \mathbf{D} \arrow[dl, "\gamma_1"'] \arrow[dd, "\gamma_1 \wedge \gamma_2"] \arrow[dr, "\gamma_2"] & \\
\mathbf{V}_1 \arrow[dr, "\lambda\langle\Gamma_1, \Gamma_1 \wedge \Gamma_2\rangle"'] & & \mathbf{V}_2 \arrow[dl, "\lambda\langle\Gamma_2, \Gamma_1 \wedge \Gamma_2\rangle"] \\
 & \mathbf{V}_1 {}_{\gamma_1}\!\wedge_{\gamma_2} \mathbf{V}_2 &
\end{tikzcd}$$

Figure 2. Relative views.

regarded as a view of Γ_1, knowledge of the precise state of the view of Γ_2 is not necessary.

For a concrete example, identify $\mathbf{D}$ with the schema $\mathbf{E}_2$ of Examples 1.2 and 3, identify Γ_1 with Π_{ABCE}, and identify Γ_2 with Π_{ABCD}. Then, as sketched in example 1.2, $\mathbf{V}_{1\,\gamma_1}\!\wedge_{\gamma_2}\mathbf{V}_2 = \Pi_{ABC}$. The mapping $\lambda\langle\Pi_{ABCE}, \Pi_{ABC}\rangle$ is the projection $\pi_{ABC} : S[ABCE] \to S[ABC]$. Similarly, $\lambda\langle\Pi_{ABCD}, \Pi_{ABC}\rangle$ is the projection $\pi_{ABC}S[ABCD] \to S[ABC]$. On the other hand, $\gamma_1 \wedge \gamma_2$ is the projection $\pi_{ABC} : S[ABCDE] \to S[ABC]$. More generally, in the context of relational schemata and views defined by projection, a pair of views forms a meet-complementary pair iff the decomposition is both lossless and dependency preserving [20, 2.17]. In this case, the meet view is just the projection on the common columns. To obtain an example in which the views form a subdirect complementary pair but not a meet complementary pair, it suffices to consider an example which is lossless but not dependency preserving, as sketched in 1.3.

In [17], the connection between decompositions of database schemata and commuting congruences is investigated thoroughly.

Summary 2.2 (The order-based framework). Despite its simplicity and elegance, the set-based framework for closed update strategies has a substantial shortcoming; namely, the update strategy depends upon the choice of the complement. For example, let $\mathbf{F}_0$ be the relational schema with the single relation $R[ABC]$, constrained by the single FD $B \to C$, and let Π_{AB} be the view defined by the projection mapping π_{AB}. Define Π_{BC} similarly. Since the pair $\{\Pi_{AB}, \Pi_{BC}\}$ forms a lossless and dependency-preserving decomposition of $\mathbf{F}_0$, it also forms a meet-complementary pair [20, 2.17]. Indeed, Π_{BC} is the 'natural' complement of Π_{AB}, and the one which yields the 'obvious' strategy for reflecting updates to Π_{AB} back to $\mathbf{F}_0$. However, as shown in [20, 1.3], it is possible to find other complements of Π_{AB} which have exactly the same meet, and so support exactly the same updates to Π_{AB}. Although these alternate complements are a bit pathological, the set-based theory outlined above in Summary 2.1 does not prefer Π_{BC} to them in any way.

To formalize this preference, additional structure must be incorporated into the model. Most database models incorporate some sort of order structure. In the relational model, the databases may be ordered via relation-by-relation inclusion. Furthermore, the common database mappings built from projection, selection, and join are all order preserving with respect to this natural order structure. In particular, while the views Π_{AB} and Π_{BC} are order mappings, the alternate views identified in [20, 2.17] are not.

The theory developed in [20] provides a systematic extension to the results outlined in Summary 2.1 above to the order-based setting. An *order schema* $\mathbf{D}$ is taken to be a poset $(\mathsf{LDB}(\mathbf{D}), \leqslant_{\mathrm{D}})$. An *order database* mapping $f : \mathbf{D}_1 \to \mathbf{D}_2$ is an order-preserving function; *i.e.*, $M_1 \leqslant_{\mathrm{D1}} M_2$ implies $f(M_1) \leqslant_{\mathrm{D2}} f(M_2)$. An *order view* $\Gamma = (\mathbf{V}, \gamma)$ of $\mathbf{D}$ consists of an order schema $\mathbf{V}$ and an open surjection $\gamma : \mathsf{LDB}(\mathbf{D}) \to \mathsf{LDB}(\mathbf{V})$; that is, a surjection which is order preserving and, in addition, which satisfies the property that whenever $N_1 \leqslant_{\mathbf{V}} N_2$, there are $M_1, M_2 \in \mathsf{LDB}(\mathbf{D})$ with the properties that $M_1 \leqslant_{\mathrm{D}} M_2, f(M_1) = N_1$, and $f(M_2) = N_2$. For the pair of order views $\{\Gamma_1, \Gamma_2\}$ to form a subdirect complementary pair in the order sense, the mapping $\gamma_1 \otimes \gamma_2 : \mathbf{D} \to \mathbf{V}_{1\,\gamma_1}\!\otimes_{\gamma_2}\mathbf{V}_2$ must be an order isomorphism (i.e., an order database mapping which is an isomorphism), and not merely an order-preserving bijection. To obtain a closed update strategy in the oder-bases sense, the formal conditions which apply in the set-based framework must be augmented with special conditions which guarantee that the update strategy respects the order structure [20, 3.1]. Modulo these modifications, it is fair to say, at least in a general way, that [20] extends the classical set-based constant complement theory to the order-based setting. As a rich source of classical but important examples, all SPJR-mappings (Select, Project, Join, Rename) in the classical relational setting define order views [20, 2.5]. The results of the current paper are cast within a more restrictive order-based context in which selection and projection, but not join, define views.

In [18], a theory of *direct* decomposition (*i.e.*, situations in which the views are independent and so the meet is trivial) of order-based schemata is presented.

3. PF-schemata and PF-views

The database schemata with order of [20] and summarized in Summary 2.2 model only those databases which satisfy the constraints of the schema; those which fail to satisfy those constraints are simply not part of the formalism. In order to model the complexity of constraint checking, it is necessary to employ a formalism which includes all databases, including those which fail to satisfy those constraints. Furthermore, to have a basis for measuring the complexity, the extended formalism must provide some notion of size of a database.

In this section, the foundations of such a framework are presented. Although it has much in common with the earlier work [21], it differs in a very fundamental way. Specifically, in [21], the starting point was the so-called semantic schemata – those which include only the databases which satisfy the constraints of the schema. From the semantic schemata the more general syntactic schemata – those which embody all databases, regardless of whether or not they satisfy the constraints – were constructed, using a sort of 'completion' operation. While this approach is effective, it is also somewhat limiting, in that it presupposes sufficient structure on the semantic schemata and their morphisms to yield unique and well-defined completions. This requirement is, unfortunately, not always met in practical examples. In the approach taken here, the starting point is the syntactic schema, with the semantic properties imposed later. As will be seen, this allows a more general class of constraints to be modelled, since unique completions are no longer required.

Definition 3.1 (CFA-schemata and morphisms). In order to capture the complexity of checking constraint satisfaction, the formal notion database schema must contain two pieces of information which are not present in the general order-based schemata of [20]. First of all, the order schema model of [20] contains only the legal databases; the idea that there are also databases which do not satisfy the constraints is not represented. Second, there is no measure of database size, without which there is no possibility of measuring complexity. To address these issues, in [21] the notions of CFA-schema and CFA-morphism were introduced. In the current paper, these notions have been replaced by the more flexible PF-schema and PF-morphism, respectively. The purpose of this subsection is to tie the 'CFA' notions to their newer 'PF' counterparts, and may safely be skipped by the reader not interested in the evolution of these ideas.

In [21], the set of all databases, legal or not, was modelled by $\mathscr{P}_f(X)$ for some (not necessarily finite) set X, while the legal databases were modelled by a special type of sub-poset of $\mathscr{P}_f(X)$, called a CFA-poset. Clearly, choosing the databases to be finite sets gives a simple measure of size – the cardinality of the set. Furthermore, taking the legal databases to be a subset of the set of all databases is completely natural and obvious. In the context of the classical relational model, think of X as the set of all possible tuples of the relation(s).

Formally, a *CFA-poset* over X is a sub-poset of $\mathscr{P}_f(X)$ which contains all singleton sets, as well as the empty set. This choice was made for the model of a database schema because it facilitated the definition of database morphisms within this context. Specifically, if $\mathbf{P} = (P, \subseteq)$ is a CFA-poset, define $\mathsf{Atoms}(\mathbf{P}) = \{\{x\} \mid x \in \bigcup P\}$, $\mathsf{ExtAtoms}(\mathbf{P}) = \mathsf{Atoms}(\mathbf{P}) \bigcup \{\emptyset\}$, and $\mathsf{Foundation}(\mathbf{P}) = \cup\, \mathsf{Atoms}(\mathbf{P})$. For any $M \in P$, the *basis* of M is $\mathsf{Basis}_{\mathbf{P}}(M) = \{\{x\} \mid x \in M\}$. A *CFA-schema* is just a CFA-poset. CFA-morphisms were then modelled as basis-preserving mappings; that is, if $\mathbf{P} = (P, \subseteq)$ and $\mathbf{Q} = (Q, \subseteq)$ are CFA-posets, then a *CFA-morphism* is a function $f : P \to Q$ with the property that, for all $M \in P$, $\{f(a) | a \in \mathsf{Basis}_{\mathbf{P}}(M)\} \setminus \{\emptyset\} = \mathsf{Basis}_{\mathbf{Q}}(f(M))$.[2] In other words, a CFA-morphism is completely defined by its action on singletons, and the image of each singleton must itself be a singleton or the empty

[2] Unfortunately, the definition given in [21] is incorrect. The condition that $\{f(a)|\ a \in \mathsf{Basis}_{\mathbf{P}}(M)\} \setminus \{\emptyset\} = \mathsf{Basis}_{\mathbf{Q}} f(M)$, which is the one intended, is expressed incorrectly as $\bigcup\{f(a)\ a \in \mathsf{Basis}_{\mathbf{P}}(M)\} = \mathsf{Basis}_{Q}(f(M))$.

set. The advantage of this definition is that it embodies the syntactic morphism as well as the syntactic one. More precisely, a morphism $f : \mathbf{P} \to \mathbf{Q}$ may be extended to $\bar{f} : \mathscr{P}_f(\mathsf{Foundation}(\mathbf{P})) \to \mathscr{P}_f(\mathsf{Foundation}(\mathbf{Q}))$ by defining $\bar{f}(M) = \bigcup\{f(\{x\}) \mid x \in M\}$. This $\bar{f}$ is the syntactic extension of f to all databases, legal or not. (As a concrete example, think of f as specifying a projection on the legal relations of a universal schema. $\bar{f}$ extends f to all relations, regardless of whether or not they satisfy the constraints.)

Definition 3.2 (PF-schemata). In a CFA-schema, all singletons, as well as the empty set, are legal databases. Translated to a universal relational schema, this means that the empty relation and all relations containing just one tuple are legal. If all constraints are typed and universal (*e.g.*, equality generating dependencies and full implicational dependencies), this condition is automatically satisfied. However, relational schemata governed by dependencies involving existential quantification, such as embedded join dependencies, inclusion dependencies, and even foreign-key dependencies, will not have this property. Since the scope of the work here is to include specifically such flavors of constraints, the model of CFA-schema must be augmented. The trick is to include two pieces of information with each schema. The first provides the equivalent of the foundation of a CFA-schema, and the second gives the legal databases, taken as a subset of the finite powerset over that foundation. While this may seem to be a bit heavy handed, remember that this is a mathematical model, not a blueprint for implementation. It is the most convenient means of recapturing all of the information necessary.

(a) A *PF-schema* is an ordered pair $\mathbf{D} = (\mathsf{SynFnd}(\mathbf{D}), \ \mathsf{LDB}(\mathbf{D}))$, in which $\mathsf{SynFnd}(\mathbf{D})$ is a nonempty set (not necessarily finite), with $\mathsf{LDB}(\mathbf{D}) \subseteq \mathscr{P}_f(\mathsf{SynFnd}(\mathbf{D}))$. (The prefix *PF*- is derived from the notation $\mathscr{P}_f(-)$.) The set $\mathsf{SynFnd}(\mathbf{D})$ is called the *syntactic foundation* of $\mathbf{D}$, and $\mathsf{LDB}(\mathbf{D})$ is the set of *legal databases* of $\mathbf{D}$.

Note that there is no requirement that each $x \in \mathsf{SynFnd}(\mathbf{D})$ occur in some $M \in \mathsf{LDB}(\mathbf{D})$, much less that $\{x\} \in \mathsf{LDB}(\mathbf{D})$ and $\emptyset \in \mathsf{LDB}(\mathbf{D})$ hold, so that the limitations of CFA-posets do not apply. In the context of a classical one-relation schema, think of $\mathsf{SynFnd}(\mathbf{D})$ as the set of tuples which the relation may contain, and $\mathsf{LDB}(\mathbf{D})$ as the set of relations over those tuples which satisfy the constraints of the schema.

For the rest of this subsection, assume that $\mathbf{D}$ is a PF-schema. There is not one but three order relations associated with $\mathbf{D}$.

(b) The *poset of legal databases* of $\mathbf{D}$ is just $\mathsf{LDBPoset}(\mathbf{D}) = (\mathsf{LDB}(\mathbf{D}), \subseteq)$

Note that $\mathsf{LDBPoset}(\mathbf{D})$ is a database schema with order, in the sense of [20], although it is not necessarily a CFA-schema in the sense of [21], since it need not contain all singletons and the empty set. It is the syntactic foundation poset which contains the representation of the latter.

(c) The *extended syntactic foundation* of $\mathbf{D}$ is $\mathsf{ExtSynFnd}(\mathbf{D}) = \mathsf{SynFnd}(\mathbf{D}) \cup \{\eta_{\mathrm{D}}\}$, in which η_{D} is a special symbol, called the *null element* of $\mathbf{D}$, which does not occur in $\mathsf{SynFnd}(\mathbf{D})$.

(d) The *syntactic foundation poset* of $\mathbf{D}$ is $\mathsf{SFPoset}(\mathbf{D}) = (\mathsf{ExtSynFnd}(\mathbf{D}), \leqslant_{\mathsf{SF}(\mathbf{D})})$. The ordering $\leqslant_{\mathsf{SF}(\mathbf{D})}$ is defined to be that in which $\eta_{\mathrm{D}} \leqslant_{\mathsf{SF}(\mathbf{D})} x$ for all $x \in \mathsf{SynFnd}(\mathbf{D})$, but is otherwise flat.

In other words, $x \leqslant_{\mathsf{SF}(\mathbf{D})} y$ holds iff $x = \eta_{\mathrm{D}}$ or else $x = y$. $\mathsf{SFPoset}(\mathbf{D})$ is trivially a database schema with order. Think of η_{D} as a special marker which represents the empty set as a database. Its rôle will become apparent later, in Definition 3.4 for the definition of a PF-morphism and in Proposition 3.9 for the modelling of an atomic equivalence relation.

The third poset is the disjoint union of the first two. Its use is purely technical; it makes it possible to consider a PF-schema as an order schema in the sense of [20].

(e) Define $\mathsf{ExtSynFnd}(\mathbf{D}) \boxplus \mathsf{LDB}(\mathbf{D})$ to be the disjoint union of these two sets, and define the order $\leqslant_{\boxplus(\mathbf{D})}$ on $\mathsf{ExtSynFnd}(\mathbf{D}) \boxplus \mathsf{LDB}(\mathbf{D})$ to be the (disjoint) union of $\subseteq$ on $\mathsf{LDB}(\mathbf{D})$ and $\leqslant_{\mathsf{SF}(\mathbf{D})}$ on $\mathsf{ExtSynFnd}(\mathbf{D})$.

(f) Define the *combined poset* of $\mathbf{D}$ to be $\mathsf{DJPoset}(\mathbf{D}) = (\mathsf{ExtSynFnd}(\mathbf{D}) \boxplus \mathsf{LDB}(\mathbf{D}), \leqslant_{\boxplus(\mathbf{D})})$.

$\mathsf{DJPoset}(\mathbf{D}) = (\mathsf{ExtSynFnd}(\mathbf{D}) \boxplus \mathsf{LDB}(\mathbf{D}), \leqslant_{\boxplus(\mathbf{D})})$ is clearly a database schema with order, in the sense of [20]. It is not necessarily a CFA-poset, although it embodies the information necessary to extend $\mathsf{LDBPoset}(\mathbf{D})$ uniquely to a CFA-poset.

The syntactic schema of $\mathbf{D}$, defined below, is constructed directly from the extended syntactic foundation; its elements represent all databases, legal or not. Note that this schema is itself a PF-schema, in the sense of (a) above, and all properties associated with PF-schemata apply to it as well.

(g) For $\mathbf{D} = (\mathsf{SynFnd}(\mathbf{D}), \mathsf{LDB}(\mathbf{D}))$, define $\overline{\mathbf{D}} = (\mathsf{SynFnd}(\mathbf{D}), \mathsf{DB}(\mathbf{D}))$ to have $\mathsf{DB}(\mathbf{D}) = \mathscr{P}_f(\mathsf{SynFnd}(\mathbf{D}))$. $\overline{\mathbf{D}}$ is called the *syntactic schema* over which $\mathbf{D}$ is taken.

To avoid confusion, when $\emptyset$ is considered as an element of $\mathsf{LDB}(\mathbf{D})$ or $\mathsf{DB}(\mathbf{D})$, it will sometimes be written as $\perp_{\mathbf{D}}$ and $\perp_{\overline{\mathbf{D}}}$, respectively.

Finally, for a database M, it is occasionally advantageous to be able to augment M with the null element for $\mathbf{D}$. This augmentation has no special semantics, but is useful in establishing certain results in which the empty set behaves like an atom.

(h) For $M \in \mathsf{DB}(\mathbf{D})$, Define $\mathsf{NullExt}(M) = M \cup \{\eta_{\mathrm{D}}\}$.

Example 3.3 (PF-schema). Let $\mathbf{E}_4$ be the two-relation schema consisting of $R_1[ABC]$ and $R_2[DEF]$. Define $\mathcal{F}_4 = \{A \to B, D \to EF\}$, and assume that these constraints hold on the schema. $\mathsf{SynFnd}(\mathbf{E}_4)$ consists of all tuples over these relations; that is, it consists of all tuples on $\mathsf{Dom}(A) \times \mathsf{Dom}(B) \times \mathsf{Dom}(C)$, together with all tuples on $\mathsf{Dom}(D)\times \mathsf{Dom}(E) \times \mathsf{Dom}(F)$. These tuples must somehow be tagged, so that it is known to which relation they belong. In this example, domain elements will be represented by (possibly subscripted) lower case letters which match the domain name; so, for example, (a_1, b_1, c_1) is associated with R_1, while (d_1, e_1, f_1) is associated with R_2. Thus, the tagging is implicit in the naming convention.

The set $\mathsf{SynFnd}(\mathbf{E_4})$ is then just the set of all such tuples. The extended syntactic foundation $\mathsf{ExtSynFnd}(\mathbf{E}_4)$ adds one additional element, which would be called η_{E_4} in the notation of Definition 3.2. The ordering on the syntactic foundation poset $\mathsf{SFPoset}(\mathbf{E}_4)$ has η_{E_4} at the bottom, with all tuples otherwise incomparable.

The set $\mathsf{DB}(\mathbf{E}_4)$ is the finite powerset of $\mathsf{SynFnd}(\mathbf{E}_4)$; that is, the set of all finite subsets of tuples, while $\mathsf{LDB}(\mathbf{E}_4)$ is the subset of $\mathsf{DB}(\mathbf{E}_4)$ consisting of sets which satisfy the constraints in $\mathcal{F}_4$. Note that in this formalism it is necessary to collect the set of all tuples of a database instance into a single set, rather than the more common practice of having a separate set of tuples for each relation. Since the tuples are tagged, this amounts to an inessential syntactic variation which has no impact upon the underlying theory.

Definition 3.4 (PF-morphisms). In the same manner that PF-schemata PF-schemata generalize CFA-schemata, PF-morphisms generalize CFA-morphisms. In the PF-context, it is no longer possible to stipulate the basis-preserving property directly on the mappings between legal databases, because the basis itself may not be embedded in those states. Rather, the starting point for a PF-morphism is a mapping between the extended syntactic foundations, which replace the bases of the CFA-context. The mapping between legal databases is then defined as an extension of the mapping between the foundations.

To begin, it is necessary to formalize the notion of an elementary mapping, as well as its extensions. For the remainder of this subsection, let $\mathbf{D}_1 = (\mathsf{SynFnd}(\mathbf{D}_1), \mathsf{LDB}(\mathbf{D}_1))$ and $\mathbf{D}_2 = (\mathsf{SynFnd}(\mathbf{D}_2), \mathsf{LDB}(\mathbf{D}_2))$ be PF-schemata.

(a) An *elementary mapping* is a function $e : \mathsf{ExtSynFnd}(\mathbf{D}_1) \to \mathsf{ExtSynFnd}(\mathbf{D}_2)$ with the property that $e(\eta_{\mathrm{D}_1}) = \eta_{\mathrm{D}_2}$.

(b) The *full extension* of the elementary mapping e is the function $e^{+} : \mathsf{DB}(\mathbf{D}_1) \to \mathsf{DB}(\mathbf{D}_2)$ given on elements by $M \mapsto \{e(x) \mid x \in\} \setminus \{\eta_{\mathrm{D}_2}\}$.

A few remarks are appropriate at this point, regarding how null elements are involved in the above definitions. In the definition of CFA-morphism in Definition 3.1, a basis preserving morphism was required to map atoms to extended atoms; that is, each atom was mapped either to another atom or else to the null set. However, the members of $\mathsf{SynFnd}(\mathbf{D}_i)$ are foundation elements, and not atoms (*i.e.*, they are of the form t, not $\{t\}$). Therefore, it is necessary to employ the special marker η_{D_i} as a sort of surrogate for $\emptyset = \{\}$ with one set of brackets stripped away. More concretely, observe that

$e : t \mapsto \eta_{\mathrm{D}_2}$ means that $e^{+} : \{t\} \mapsto \emptyset = \bot_{\mathbf{D}_2}$. Note also that the null elements are always stripped away in the full extension, since that function deals with databases, of which a null element is never a member.

It may also be remarked that an elementary mapping could have been defined to be of the form $e : \mathsf{SynFnd}(\mathbf{D}_1) \to \mathsf{ExtSynFnd}(\mathbf{D}_2)$, since null elements are not needed in the domain. However, this would have complicated the definition of morphism composition somewhat. Including η_{D_1} in the domain of e is harmless enough, since it is always mapped to η_{D_2}.

Next, the details of extending an elementary mapping to mappings between databases are elaborated.

(c) The elementary mapping e is *semantically stable* if $e^{+}(\mathsf{LDB}(\mathbf{D}_1)) \subseteq \mathsf{LDB}(\mathbf{D}_2)$; *i.e.*, for each $M \in \mathsf{LDB}(\mathbf{D}_1)$, $e^{+}(M) \in \mathsf{LDB}(\mathbf{D}_2)$.

(d) Formally, a *PF-morphism* $f : \mathbf{D}_1 \to \mathbf{D}_2$ is a triple $(f^{\sharp}, f^{\flat}, f^{\boxplus})$ in which

  (i) $f^{\sharp} : \mathsf{ExtSynFnd}(\mathbf{D}_1) \to \mathsf{ExtSynFnd}(\mathbf{D}_2)$ is a semantically stable elementary mapping;

  (ii) $f^{\flat} : \mathsf{LDB}(\mathbf{D}_1) \to \mathsf{LDB}(\mathbf{D}_2)$ with $f^{\flat}(M) = (f^{\sharp})^{+}(M)$ for all $M \in \mathsf{LDB}(\mathbf{D})$.

  (iii) $f^{\boxplus} : \mathsf{ExtSynFnd}(\mathbf{D}_1) \boxplus \mathsf{LDB}(\mathbf{D}_1) \to \mathsf{ExtSynFnd}(\mathbf{D}_2) \boxplus \mathsf{LDB}(\mathbf{D}_2)$ given by $f^{\boxplus}(x) = f^{\sharp}(x)$ if $x \in \mathsf{SynFnd}(\mathbf{D}_1)$, and $f^{\boxplus}(x) = f^{\flat}(x)$ if $x \in \mathsf{LDB}(\mathbf{D}_1)$.

This definition may seem to be needlessly complex, since f is completely determined by $f^{\sharp}$, and $f^{\boxplus}$ is obtained by combining $f^{\sharp}$ and $f^{\flat}$. However, a notation is needed for all three components in any case, and including them explicitly in the definition causes no harm, while making the definition of morphism composition more concrete.

Composition of morphisms is defined in the obvious way; that is, $f \circ g$ is given by the triple $(f^{\sharp} \circ g^{\sharp}, f^{\flat} \circ g^{\flat}, f^{\boxplus} \circ g^{\boxplus})$.

It is clear that all three of $f^{\sharp} : \mathsf{SFPoset}(\mathbf{D}_1) \to \mathsf{SFPoset}(\mathbf{D}_2)$, $f^{\flat} : \mathsf{LDBPoset}(\mathbf{D}_1) \to \mathsf{LDBPoset}(\mathbf{D}_2)$, and $f^{\boxplus} : \mathsf{DJPoset}(\mathbf{D}_1) \to \mathsf{DJPoset}(\mathbf{D}_2)$ may be viewed as poset morphisms. $f^{\boxplus}$ embodies the ideas of CFA-morphism, although it is not formally one itself.

The notions of a surjective morphism and an open morphism must be defined carefully here, since they focus on $\mathsf{LDB}(\mathbf{D})$. The definition of isomorphism is standard [2, 3.8].

(e) The PF-morphism f will be called *surjective* precisely in the case that both $f^{\sharp} : \mathsf{ExtSynFnd}(\mathbf{D}_1) \to \mathsf{ExtSynFnd}(\mathbf{D}_2)$ and $f^{\flat} : \mathsf{LDB}(\mathbf{D}_1) \to \mathsf{LDB}(\mathbf{D}_2)$ are surjective. (Note that this is not equivalent to the surjectivity of $f^{\sharp}$ alone.)

(f) The PF-morphism f is *open* if for any $N_1, N_2 \in \mathsf{LDB}(\mathbf{D}_2)$ with $N_1 \subseteq N_2$, there are $M_1, M_2 \in \mathsf{LDB}(\mathbf{D}_1)$ with $M_1 \subseteq M_2$ and $f^{\flat}(M_1) = N_1, f^{\flat}(M_2) = N_2$.

(g) The PF-morphism f is a *PF-isomorphism* if there is a morphism $g : \mathbf{D}_2 \to \mathbf{D}_1$ with the property that $g \circ f$ is the identity on $\mathbf{D}_1$ and $f \circ g$ is the identity on $\mathbf{D}_2$.

Finally, it is useful to have a special notation for the PF-morphism between the syntactic schemata derived from $\mathbf{D}_1$ and $\mathbf{D}_2$.

(h) Define the PF-morphism $\bar{f} : \overline{\mathbf{D}}_1 \to \overline{\mathbf{D}}_2$ as $(\bar{f}^\sharp, \bar{f}^\flat, \bar{f}^\boxplus)$, with $\bar{f}^\sharp = \bar{f}$, $\bar{f}^\flat = f^{\sharp +}$, and $\bar{f}^\boxplus$ given by $\bar{f}^\boxplus(x) = f^\sharp(x)$ if $x \in \mathsf{SynFnd}(\mathbf{D}_1)$, and $\bar{f}^\boxplus(x) = \bar{f}^\flat(x)$ if $x \in \mathsf{DB}(\mathbf{D_1})$.

Example 3.5 (PF-morphisms). The example builds upon the context introduced in example 3.3. Let $\mathbf{W}1_{AB}$ be the schema whose sole relation is $R_3[AB]$, constrained by the FD $A \to B$. Define the elementary mapping $h_4 : \mathsf{ExtSynFnd}(\mathbf{E}_4) \to \mathsf{ExtSynFnd}(\mathbf{W}1_{AB})$ by $(a, b, c) \mapsto (a, b)$ if $(a, b, c) \in \mathsf{Dom}(A) \times \mathsf{Dom}(B) \times \mathsf{Dom}(C)$, $(d, e, f) \mapsto \eta_{\mathbf{W}1_{AB}}$ if $(d, e, f) \in \mathsf{Dom}(D) \times \mathsf{Dom}(E) \times \mathsf{Dom}(F)$, and $\eta_{\mathbf{E}_4} \mapsto \eta_{\mathbf{W}1_{AB}}$.

The full extension $h_4^+ : \mathsf{DB}(\mathbf{E}_4) \to \mathsf{DB}(\mathbf{W}1_{AB})$ is the morphism which projects each tuple of the instance of $R_1[ABC]$ onto its AB projection, and ignores all tuples in the instance of $R_2[DEF]$, while $h_4^\flat : \mathsf{LDB}(\mathbf{E}_4) \to \mathsf{LDB}(\mathbf{W}1_{AB})$ exhibits the same behavior on the instances satisfying $\mathcal{F}_4$. Thus, if $M = \{(a_1, b_1, c_1), a_2, b_2, c_2), (d_1, e_1, f_1), (d_2, e_2, f_2)\}$, then $\omega1_{AB}(M) = \{(a_1, b_1), (a_2, b_2)\}$. The morphism h_4 is clearly semantically stable; however, if the constraint $A \to B$ were dropped on $\mathbf{E}_4$, with that same constraint retained on $R_3[AB]$, then semantic stability would no longer hold.

It is important to note in particular the need for the null elements $\eta_{\mathbf{W}1_{AB}}$. Without it, it would not be possible to specify the behavior of $\omega1_{AB}$ on an element-by-element basis, since it would then be impossible to specify the image of tuples in the instance of $R_2[DEF]$.

Discussion 3.6 (The limitations of PF-morphisms). In a PF-morphism, the mapping is defined element by element. Thus, in the context of the classical relational model, a PF-morphism can represent operations such as projection, selection, and renaming. However, join cannot be so represented, since the join operation involves the combination of two tuples. This may seem to be a rather severe limitation, but it is necessary to obtain the results developed in this work. Element-by-element operators, such as projection and restriction, do not generate any constraints themselves; rather, they simply pass along information about the constraints which already exist. On the other hand, an operation such as join imposes a constraint (a join dependency) itself on the image, a constraint which may not exist on the domain. Thus, a situation in which a single element in the view schema is constructed from several elements in the main schema complicates matters enormously. The view mapping may itself impose constraints, so that constraints now come from two sources, the other being the implied constraints from the main schema. It seems most prudent here to begin with PF-morphisms, and to look for extensions only after the theory for these element-by-element mappings is understood better.

Definition 3.7. (PF-views). The abstract definition of a PF-view parallels closely that of an order view [20, 2.3] and a CFA-view [21, 3.2]. Specifically, let $(\mathsf{SynFnd}(\mathbf{D}), \mathsf{LDB}(\mathbf{D}))$ be a PF-schema.

(a) A *PF-view* of $\mathbf{D}$ is a pair $\Gamma = (\mathbf{V}, \gamma)$ in which $\mathbf{V} = (\mathsf{SynFnd}(\mathbf{V}), \mathsf{LDB}(\mathbf{V}))$ is a PF-schema and $\gamma : \mathbf{D} \to \mathbf{V}$ is an open surjective PF-morphism.

Note that *open* and *surjective* have the meanings assigned in Definition 3.4(e)–(f) above. Furthermore, it is easy to see that $\Gamma_1 = (\mathbf{V}_1, \gamma_1)$ and $\Gamma_2 = (\mathbf{V}_2, \gamma_2)$ of $\mathbf{D}$ are isomorphic if there is a PF-isomorphism $f : \mathbf{V}_1 \to \mathbf{V}_2$ with the property that $f \circ \gamma_1 = \gamma_2$. Indeed, if f is a PF-isomorphism, then $f^{-1} \circ \gamma_2 = f^{-1} \circ (f \circ \gamma_1) = (f^{-1} \circ f) \circ \gamma_1 = \gamma_1$.

In the context of updates, it is important to have available the associated syntactic view, which is defined as follows.

(b) The *extension of* Γ *to* $\overline{\mathbf{D}}$ is $\overline{\Gamma} = (\overline{\mathbf{V}}, \overline{\gamma})$.

It is straightforward to verify that $\overline{\Gamma}$ is a PF-view of $\overline{\mathbf{D}}$.

Discussion 3.8 (Congruences on a PF-schema). The definition of closed update strategy is made within the context of meet-complementary views, and the definition of the latter is based centrally upon the congruences of those views. It is therefore critical to classify these congruences and identify their rôles in the construction of complementary and meet-complementary views. Just as a morphism has three functional components, so too does a view have three congruences. In addition, the congruence of the syntactic extension $\bar{f}$ is also identified explicitly, since it is used in subsequent proofs.

Let $\mathbf{D}_1 = (\mathsf{SynFnd}(\mathbf{D}_1), \mathsf{LDB}(\mathbf{D}_1))$ and $\mathbf{D}_2 = (\mathsf{SynFnd}(\mathbf{D}_2), \mathsf{LDB}(\mathbf{D}_2))$ be PF-schemata, and let $f : \mathbf{D}_1 \to \mathbf{D}_2$ be a PF-morphism.

(a) The *elementary congruence* on $\mathbf{D}_1$ *induced by* f is the relation $\mathsf{EltCongr}(f)$ on $\mathsf{ExtSynFnd}(\mathbf{D}_1)$ given by $(x, y) \in \mathsf{EltCongr}(f)$ iff $f^{\sharp}(x) = f^{\sharp}(y)$.

(b) The *full syntactic congruence* on $\mathbf{D}_1$ *induced by* f is the relation $\mathsf{SynCongr}(f)$ on $\mathsf{DB}(\mathbf{D})$ given by $(M_1, M_2) \in \mathsf{SynCongr}(f)$ iff $\bar{f}^{\flat}(M_1) = \bar{f}^{\flat}(M_2)$.

(c) The *full semantic congruence* on $\mathbf{D}_1$ *induced by* f is the relation $\mathsf{SemCongr}(f)$ on $\mathsf{LDB}(\mathbf{D})$ given by $(M_1, M_2) \in \mathsf{SemCongr}(f)$ iff $f^{\flat}(M_1) = f^{\flat}(M_2)$.

(d) The *full congruence* on $\mathbf{D}_1$ induced by f is the relation $\mathsf{Congr}(f)$ on $\mathsf{ExtSynFnd}(\mathbf{D}_1) \boxplus \mathsf{LDB}(\mathbf{D}_1)$ given by $(P_1, P_2) \in \mathsf{Congr}(f)$ iff $f^{\boxplus}(P_1) = f^{\boxplus}(P_2)$.

These four congruences have natural counterparts on views. Let $\Gamma = (\mathbf{V}, \gamma)$ be an PF-view of $\mathbf{D}$.

(e) The *elementary congruence* of Γ, denoted $\mathsf{EltCongr}(\Gamma)$, is just $\mathsf{EltCongr}(\gamma)$.

(f) The *syntactic congruence* of Γ, denoted $\mathsf{SynCongr}(\Gamma)$, is just $\mathsf{SynCongr}(\gamma)$.

(g) The *semantic congruence* of Γ, denoted $\mathsf{SemCongr}(\Gamma)$, is just $\mathsf{SemCongr}(\gamma)$.

(h) The *full congruence* of Γ, denoted $\mathsf{Congr}(\Gamma)$, is just $\mathsf{Congr}(\gamma)$.

The congruences $\mathsf{SynCongr}(f)$ and $\mathsf{SemCongr}(f)$ have a special structure. Since they are defined in terms of $\mathsf{EltCongr}(f)$, two sets are equivalent if and only if each element in one is equivalent to an element in the other. In the formalization below, note in particular the use of $\mathsf{NullExt}(-)$ (see Definition 3.2) to support the fact that the syntactic foundation $f^\sharp$ of a PF-morphism f involves the null element as well as the elements of the syntactic foundation.

Proposition 3.9 (Characterization of equivalences). Let $\mathbf{D}_1 = (\mathsf{SynFnd}(\mathbf{D}_1), \mathsf{LDB}(\mathbf{D}_1))$ and $\mathbf{D}_2 = (\mathsf{SynFnd}(\mathbf{D}_2), \mathsf{LDB}(\mathbf{D}_2))$ be PF-schemata, and let $f : \mathbf{D}_1 \to \mathbf{D}_2$ be a PF-morphism.

(a) For $M_1, M_2 \in \mathsf{DB}(\mathbf{D})$, $(M_1, M_2) \in \mathsf{SynCongr}(f)$ iff the following two conditions are satisfied:

(eeq-i) $(\forall x \in M_1)(\exists y \in \mathsf{NullExt}(M_2))((x, y) \in \mathsf{EltCongr}(f))$.

(eeq-ii) $(\forall y \in M_2)(\exists x \in \mathsf{NullExt}(M_1)((x, y) \in \mathsf{EltCongr}(f))$.

(b) Similarly, for $M_1, M_2 \in \mathsf{LDB}(\mathbf{D})$, $(M_1, M_2) \in \mathsf{SemCongr}(f)$ iff the conditions (eeq-i) and (eeq-2) are satisfied.

Proof. Part (a) follows from the fact that $\bar{f}$ is defined to be the full extension of $f^\sharp$ (see Definition 3.4). Part (b) then follows immediately, since $\mathsf{SemCongr}(f) \subseteq \mathsf{SynCongr}(f)$. □

Definition 3.10 (Null-augmented congruence containment). Given PF-views $\Gamma_1 = (\mathbf{V}_1, \gamma_1)$ and $\Gamma_2 = (\mathbf{V}_2, \gamma_2)$ of the PF-schema $\mathbf{D}$, to say that the congruence of Γ_1 is finer than that of Γ_2¡ it does not suffice to assert only that $\mathsf{Congr}(\Gamma_1) \subseteq \mathsf{Congr}(\Gamma_2)$. It is also necessary to guarantee that whenever ${\gamma^\sharp}_1$ maps a particular element to the null element, so too does ${\gamma^\sharp}_2$. More precisely, proceed as follows.

(a) Define $\mathsf{EltCongr}(\Gamma_1) \Subset \mathsf{EltCongr}(\Gamma_2)$ to mean that $(\forall x \in \mathsf{SynFnd}(\mathbf{D}_1))(({\gamma^\sharp}_1(x) = \eta_{\mathbf{V}_1}) \Rightarrow ({\gamma^\sharp}_2(x) = \eta_{\mathbf{V}_2}))$.

When this condition is appended to a usual set-theoretic containment of congruences, new definitions, denoted using $\sqsubseteq$ instead of $\subseteq$, are obtained.

(b) Define $\mathsf{EltCongr}(\Gamma_1) \sqsubseteq \mathsf{EltCongr}(\Gamma_2)$ to mean that both $\mathsf{EltCongr}(\Gamma_1) \subseteq \mathsf{EltCongr}(\Gamma_2)$ and $\mathsf{EltCongr}(\Gamma_1) \Subset \mathsf{EltCongr}(\Gamma_2)$ hold.

(c) Define $\mathsf{SynCongr}(\Gamma_1) \sqsubseteq \mathsf{SynCongr}(\Gamma_2)$ to mean that both $\mathsf{SynCongr}(\Gamma_1) \subseteq \mathsf{SynCongr}\ (\Gamma_2)$ and $\mathsf{EltCongr}(\Gamma_1) \mathrel{\underline{\overset{+}{\subseteq}}} \mathsf{EltCongr}(\Gamma_2)$ hold.

(d) Define $\mathsf{Congr}(\Gamma_1) \sqsubseteq \mathsf{Congr}(\Gamma_2)$ to mean that both $\mathsf{Congr}(\Gamma_1) \subseteq \mathsf{Congr}(\Gamma_2)$ and $\mathsf{EltCongr}(\Gamma_1) \mathrel{\underline{\overset{+}{\subseteq}}} \mathsf{EltCongr}(\Gamma_2)$ hold.

Next, the conditions which ensure that the fill-in of figure 1 exists in the framework of PF-morphisms are established.

Proposition 3.11. Let $\mathbf{D} = (\mathsf{SynFnd}(\mathbf{D}), \mathsf{LDB}(\mathbf{D}))$ be a PF-schema, and let $\Gamma_1 = (\mathbf{V}_1, \gamma_1)$ and $\Gamma_2 = (\mathbf{V}_2, \gamma_2)$ be PF-views of $\mathbf{D}$. The following conditions are equivalent.

(a) $\mathsf{EltCongr}(\Gamma_1) \sqsubseteq \mathsf{EltCongr}(\Gamma_2)$.

(b) $\mathsf{SynCongr}(\Gamma_1) \sqsubseteq \mathsf{SynCongr}(\Gamma_2)$.

(c) $\mathsf{Congr}(\Gamma_1) \sqsubseteq \mathsf{Congr}(\Gamma_2)$.

(d) There is an open surjective PF-morphism $f : \mathbf{V}_1 \to \mathbf{V}_2$ with the property that $\gamma_2 = f \circ \gamma_1$.

Proof. The equivalence of conditions (a)–(c), as well as (d) $\Rightarrow$ (c), are immediate.

Suppose that conditions (a)–(c) are satisfied. Define $g : \mathsf{SynFnd}(\mathbf{V}_1) \to \mathsf{SynFnd}(\mathbf{V}_2)$ as follows. For $y \in \mathsf{SynFnd}(\mathbf{V}_1)$, pick any $x \in \mathsf{SynFnd}(\mathbf{D})$ with $\gamma^\sharp{}_1(x) = y$, and define $g(y) = \gamma^\sharp{}_2(x)$. The fact that $\mathsf{EltCongr}(\Gamma_1) \subseteq \mathsf{EltCongr}(\Gamma_2)$ ensures that this definition is independent of the particular choice of $x \in (\gamma^\sharp{}_1)^{-1}(y)$. Now, simply define f to be the unique PF-morphism with $f^\sharp = g$. The fact that $(\forall x \in \mathsf{SynFnd}(\mathbf{D}_1))((\gamma^\sharp{}_1(x) = \eta_{\mathrm{D}_1}) \Rightarrow (\gamma^\sharp{}_2(x) = \eta_{\mathrm{D}_2}))$ ensures that f will be surjective. To see that $f^\flat : \mathsf{LDB}(\mathbf{V}_1) \to \mathsf{LDB}(\mathbf{V}_2)$ is open, let $N_1, N_2 \in \mathsf{LDB}(\mathbf{V}_2)$ with $N_1 \subseteq N_2$, and choose $M_1, M_2 \in \mathsf{LDB}(\mathbf{D})$ with $M_1 \subseteq N_2$ and $\gamma_2^\flat(M_1) = N_1$, $\gamma_2^\flat(M_2) = N_2$. Then $\gamma_2^\flat(M_1) = f^\flat(\gamma_1^\flat(M_1)) \subseteq f^\flat(\gamma_1^\flat(M_2)) = \gamma_2^\flat(M_2)$, so $(\gamma_1^\flat(M_1), \gamma_2^\flat(M_2))$ is the desired pair which maps to (N_1, N_2) under $f^\flat$. $\square$

Definition 3.12 (Relative views). Let $\mathbf{D} = (\mathsf{SynFnd}(\mathbf{D}), \mathsf{LDB}(\mathbf{D}))$ be a PF-schema, and let $\Gamma_1 = (\mathbf{V}_1, \gamma_1)$ and $\Gamma_2 = (\mathbf{V}_2, \gamma_2)$ be PF-views of $\mathbf{D}$. Suppose further that the equivalent conditions of Proposition 3.11 are satisfied. The *relative view* from Γ_1 to Γ_2 is the PF-view $\Lambda(\Gamma_1, \Gamma_2) = (\mathbf{V}_2, \lambda\langle\Gamma_1, \Gamma_2\rangle)$ of $\mathbf{V}_1$, with $\lambda\langle\Gamma_1, \Gamma_2\rangle : \mathbf{V}_1 \to \mathbf{V}_2$ the unique fill-in identified in Proposition 3.11(d) above. This situation is depicted in

$$\begin{tikzcd}
 & \mathbf{D} \arrow[dl, "\gamma_1"'] \arrow[dr, "\gamma_2"] & \\
\mathbf{V}_1 \arrow[rr, "\lambda\langle\Gamma_1, \Gamma_2\rangle"] & & \mathbf{V}_2
\end{tikzcd}$$

Figure 3. Relative view morphism.

figure 3. The concept of a relative view is particularly critical to the construction of a meet, as illustrated in figure 2, and developed in 4.3.

As a concrete example, think of $\mathbf{D}$ as $R[ABC]$, $\mathbf{V}_1$ as $R[AB]$, and $\mathbf{V}_2$ as $R[B]$, with $\gamma_1 = \pi_{AB} : R[ABC] \to R[AB]$, and $\gamma_2 = \pi_B : R[ABC] \to R[B]$. Then $\lambda\langle\Gamma_1, \Gamma_2\rangle = \pi_B : R[AB] \to R[B]$.

Definition 3.13 (The view defined by an elementary congruence). In some important constructions, it is necessary to construct a view from a congruence. (In this work, the meet is such a construction; see Definition 4.3(c)) for details.) This is a rather straightforward process; the steps are outlined below.

Let $(\mathsf{SynFnd}(\mathbf{D}), \mathsf{LDB}(\mathbf{D}))$ be a PF-schema, and let R be an equivalence relation on $\mathsf{ExtSynFnd}(\mathbf{D})$. A PF-view of $\mathbf{D}$, denoted $[\![\mathbf{D}]\!]/\!\!/R$ and unique up to isomorphism, with the property that the elementary equivalence of the view is R, may be constructed via the following steps.

(a) Define $R_{\not\eta} = R_{\mid \mathsf{SynFnd}(\mathbf{D})} = R \cap (\mathsf{SynFnd}(\mathbf{D}) \times \mathsf{SynFnd}(\mathbf{D}))$.

(b) Define $\mathsf{SynFnd}([\![\mathbf{D}]\!]/\!\!/R) = \mathsf{SynFnd}(\mathbf{D})/R_{\not\eta} =$ the set of equivalence classes of SynFnd $(\mathbf{D})$ under the equivalence relation $R_{\not\eta}$.

(c) For $M \in \mathsf{DB}(\mathbf{D})$, define $[M]_{R_{\not\eta}} = \{[x]_{R_{\not\eta}} \mid x \in M \text{ and } (x, \eta_{\mathrm{D}}) \notin R\}$. ($[x]_{R_{\not\eta}}$ denotes the block of $\mathsf{SynFnd}(\mathbf{D})/R_{\not\eta}$ in which x lies.)

(d) Define $\mathsf{LDB}([\![\mathbf{D}]\!]/\!\!/R) = \{[M]_{R_{\not\eta}} \mid M \in \mathsf{LDB}(\mathbf{D})\}$.

(e) Define $[\![\mathbf{D}]\!]/\!\!/R = (\mathsf{SynFnd}([\![\mathbf{D}]\!]/\!\!/R, [\![\mathsf{LDB}(\mathbf{D})]\!]/\!\!/R)$.

It is easy to see that $[\![\mathbf{D}]\!]/\!\!/R$ is a PF-schema.

(f) Define $g : \mathsf{ExtSynFnd}(\mathbf{D}) \to \mathsf{ExtSynFnd}([\![\mathbf{D}]\!]/\!\!/R)$ on elements as follows.

$$x \mapsto \begin{cases} [x]_{R_\eta} & \text{if } (x, \eta_{\mathrm{D}}) \notin R. \\ \eta_{[\![\mathbf{D}]\!]/\mathbf{R}} & \text{if } (x, \eta_{\mathrm{D}}) \in R. \end{cases}$$

(g) Let $[\![\theta]\!]_R : \mathbf{D} \to \mathbf{D}/\!\!/R$ be the PF-morphism with $([\![\theta]\!]_R)^\sharp = g$.

(h) Finally, put $[\![\Theta]\!]_R = ([\![\mathbf{D}]\!]/\!\!/R, [\![\theta]\!]_R)$.

It is easy to see that $[\![\Theta]\!]_R$ is a PF-view which has R as its elementary congruence.

4. Decompositions in the context of PF-views

In order to be able to use PF-views as the abstract model for meet-complementary pairs, it is necessary to show that the key ideas which were developed in the earlier work [20] can be extended to this framework. Although the main ideas are essentially the same, there are a number of details to be considered, particularly as regards the congruence to use in defining complementary pairs.

Definition 4.1 (Products and complements of views). Let $\Gamma_1 = (\mathbf{V}_1, \gamma_1)$ and $\Gamma_2 = (\mathbf{V}_2, \gamma_2)$ be PF-views of the PF-schema $\mathbf{D}$. Unfortunately, there does not appear to be a clean way to extend the definition of view product presented in [20, 2.1] to the PF-context in such a way that $\gamma_1 \otimes \gamma_2$ is a PF-morphism. The problem is that the product mapping cannot be defined by an elementary mapping, since a single element in the main schema $\mathbf{D}$ maps to two elements in the decomposition, one in $\mathsf{LDB}(\mathbf{V}_1)$ and the other in $\mathsf{LDB}(\mathbf{V}_2)$. Fortunately, this is not a serious problem. There is no need to require the decomposition mapping to be a PF-morphism; it is quite sufficient that it be a poset morphism, and so the definitions of the earlier work suffice. The appropriate translations of these definitions are as follows.

(a) Define $\mathsf{LDB}(\mathbf{V}_1 {}_{\gamma_1}\!\otimes_{\gamma_2} \mathbf{V}_2) = \{(\gamma_1^\flat(M), \gamma_2^\flat(M)) \mid M \in \mathsf{LDB}(\mathbf{D})\}$.

(b) Define the function $\gamma_1^\flat \otimes \gamma_2^\flat : \mathsf{LDB}(\mathbf{D}) \to \mathsf{LDB}(\mathbf{V}_1 {}_{\gamma_1}\!\otimes_{\gamma_2} \mathbf{V}_2)$ by $M \mapsto (\gamma_1^\flat(M),\ \gamma_2^\flat(M))$.

(c) The pair $\{\Gamma_1, \Gamma_2\}$ of PF-views is said to form a *subdirect complementary pair* just in case $\gamma_1^\flat \otimes \gamma_2^\flat : \mathsf{LDB}(\mathbf{D}) \to \mathsf{LDB}(\mathbf{V}_1 {}_{\gamma_1}\!\otimes_{\gamma_2} \mathbf{V}_2)$ is a poset isomorphism, with both sets carrying the natural order defined by set inclusion $\subseteq$. In this case, it is also said that Γ_1 and Γ_2 are *subdirect complements* of one another.

In the more general case of order schemata, it is quite possible for $\gamma_1^\flat \otimes \gamma_2^\flat$ to be a bijection without being a poset isomorphism [20, 2.11]. However, in the context of PF-views, this is not possible, as is established by the following proposition.

Proposition 4.2. Let $\mathbf{D} = (\mathsf{SynFnd}(\mathbf{D}), \mathsf{LDB}(\mathbf{D}))$ be a PF-schema, and let $\{\Gamma_1, \Gamma_2\}$ be a pair of PF-views of $\mathbf{D}$. For $\{\Gamma_1, \Gamma_2\}$ to be a subdirect complementary pair it is necessary and sufficient that $\gamma_1^\flat \otimes \gamma_2^\flat$ be injective.

Proof. Since $\gamma_1^\flat \otimes \gamma_2^\flat$ is surjective by construction, under the injectivity assumption it is bijective. It is immediate that it is also a poset morphism. That $(\gamma_1^\flat \otimes \gamma_2^\flat)^{-1}$ is a poset morphism is very straightforward also. Indeed, for $M, N \in \mathsf{LDB}(\mathbf{D})$ with $\gamma_1^\flat(M) \subseteq \gamma_1^\flat(N)$ and $\gamma_2^\flat(M) \subseteq \gamma_2^\flat(N)$, it is immediate from the definition of product ordering that $M \subseteq N$. $\square$

Definition 4.3 (Fully commuting views and meet complements). The importance of commuting congruences cannot be overstated. It is the condition on a complementary pair which ensures that constant-complement update strategies are independent of the state of the complement [20, 3.10]. In addition, commuting congruences play a key role in the characterization of 'desirable' properties of decompositions which are related to the classical notion of acyclic schemata [17].

Because of the way in which the (full) congruence of a PF-view is defined, the formal definition is identical to that for order views.

(a) The pair $\{\Gamma_1, \Gamma_2\}$ of PF-views of the PF-schema $\mathbf{D}$ is called a *fully commuting pair* if $\mathsf{Congr}(\Gamma_1) \circ \mathsf{Congr}(\Gamma_2) = \mathsf{Congr}(\Gamma_2) \circ \mathsf{Congr}(\Gamma_1)$, with "$\circ$" denoting ordinary composition of binary relations, as already sketched in Summary 2.1 for set-based views. In view of the definitions of Discussion 3.8, this is equivalent to simultaneous satisfaction of the following two conditions:

(i) $\mathsf{EltCongr}(\Gamma_1) \circ \mathsf{EltCongr}(\Gamma_2) = \mathsf{EltCongr}(\Gamma_2) \circ \mathsf{EltCongr}(\Gamma_1)$.

(ii) $\mathsf{SemCongr}(\Gamma_1) \circ \mathsf{SemCongr}(\Gamma_2) = \mathsf{SemCongr}(\Gamma_2) \circ \mathsf{SemCongr}(\Gamma_1)$.

Of course, if the semantic congruences $\mathsf{SemCongr}(\Gamma_1)$ and $\mathsf{SemCongr}(\Gamma_2)$ are sufficiently rich; that is, if they contain enough information to reconstruct the corresponding elementary congruences, then condition (ii) suffices. This will be the case if the schemata and views are definable as CFA-schemata and CFA-views, but it is not true in general. This provides the motivation for the rather heavy constructions of combining the two components using the '$\boxplus$' operators.

(b) A subdirect complementary pair $\{\Gamma_1, \Gamma_2\}$ which is fully commuting is called a *meet-complementary pair*, and Γ_1 and Γ_2 are called *meet complements* of one another.

(c) If Γ_1, Γ_2 is a meet-complementary pair of PF-views of the PF-schema $\mathbf{D}$, the view (unique up to isomorphism – see 3.13) whose congruence is $\mathsf{Congr}(\Gamma_1) \circ \mathsf{Congr}(\Gamma_2)$ is called the *meet* of $\{\Gamma_1, \Gamma_2\}$, and is denoted $\Gamma_1 \wedge \Gamma_2$. It is immediate that $\mathsf{Congr}(\Gamma_i) \sqsubseteq \mathsf{Congr}(\Gamma_1 \wedge \Gamma_2)$ for $i \in \{1, 2\}$.

Definition 4.4 (Independence dependencies). The meet of a pair of meet-complementary views defines the common ground on which the views must agree. More formally, let $\mathbf{D} = (\mathsf{SynFnd}(\mathbf{D}), \mathsf{LDB}(\mathbf{D}))$ be a PF-schema, let $\{\Gamma_1, \Gamma_2\}$ be a subdirect complementary pair of PF-views, and let $\{\Gamma_1, \Gamma_2\}$ be a PF-view of $\mathbf{D}$, with $\mathsf{Congr}(\Gamma_1) \sqsubseteq \mathsf{Congr}(\Gamma_3)$ and $\mathsf{Congr}(\Gamma_2) \sqsubseteq \mathsf{Congr}(\Gamma_3)$.

(a) The Γ_3-*independence dependency* on $\mathbf{V}_{1\,\gamma_1}\!\otimes_{\gamma_2}\mathbf{V}_2$, denoted $\otimes_{\Gamma_3}$, is satisfied iff for any $M_1 \in \mathsf{LDB}(\mathbf{V}_1)$ and $M_2 \in \mathsf{LDB}(\mathbf{V}_2)$, the following condition is satisfied [20, 2.13].

$$\text{(id)} \qquad ((M_1, M_2) \in \mathsf{LDB}(\mathbf{V}_{1\,\gamma_1}\!\otimes_{\gamma_2}\mathbf{V}_2)) \iff (\lambda\langle\Gamma_1, \Gamma_3\rangle(M_1) = \lambda\langle\Gamma_2, \Gamma_3\rangle(M_2))$$

In the context of PF-schemata, this dependency may be expressed in an element-by-element fashion. To understand the difference, first consider the example of a simple relational schema $R[ABC]$ constrained by the single FD $B \to C$, which decomposes losslessly and in a dependency-preserving fashion into the two projections Π_{AB} and Π_{BC}. In this case, the meet $\Pi_{AB} \wedge \Pi_{BC} = \Pi_B$ [20, 2.17]. The Π_B-independence dependency asserts that the projection of the state of each view on attribute B is the same. The alternate, element-by-element characterization states that for each tuple (a,b) in the state of the view Π_{AB}, there is a tuple (b,c) in the state of the view Π_{BC} with a matching B-value, and conversely. These conditions are so obviously identical that it may seem pointless to differentiate between them. However, in a more general context, they display an important difference. The Π_B-dependency characterization is formulated within the very general framework of order views; no concept of tuple is necessary. On the other hand, a generalization of the tuple-by-tuple matching condition requires a corresponding abstraction of the notion of a tuple; while general order views do not support this abstraction, PF-views do. The formalizations are as follows.

(b) The *pointwise* Γ_3*-independence dependency* is satisfied iff the following two dual conditions are met for each $(M_1, M_2) \in \mathsf{LDB}(\mathbf{V}_{1\,\gamma_1}\!\otimes_{\gamma_2}\mathbf{V}_2)$.

(id : 1)
$$(\forall x_1 \in M_1)(\exists x_2 \in \mathsf{NullExt}\ (M_2))(\lambda\langle\Gamma_1,\Gamma_3\rangle^\sharp(x_1) = \lambda\langle\Gamma_2,\Gamma_3\rangle^\sharp(x_2))$$

(id : 2)
$$(\forall x_2 \in M_2)(\exists x_1 \in \mathsf{NullExt}(M_1))(\lambda\langle\Gamma_1,\Gamma_3\rangle^\sharp(x_1) = \lambda\langle\Gamma_2,\Gamma_3\rangle^\sharp(x_2))$$

In other words, in the context of PF-views, conditions (id:1) and (id:2) may replace (id).

The formalization of these ideas is recorded in the following proposition.

Proposition 4.5. Let $(\mathsf{SynFnd}(\mathbf{D}), \mathsf{LDB}(\mathbf{D}))$ be a PF-schema, let $\{\Gamma_1, \Gamma_2\}$ be a subdirect complementary pair of PF-views of $\mathbf{D}$, and let Γ_3 be the view (unique up to isomorphism) whose congruence is the smallest equivalence relation on $\mathsf{ExtSynFnd}(\mathbf{D}_1) \boxplus \mathsf{LDB}(\mathbf{D}_1)$ containing both $\mathsf{Congr}(\Gamma_1)$ and $\mathsf{Congr}(\Gamma_2)$. Then $\{\Gamma_1, \Gamma_2\}$ is a meet-complementary pair iff conditions (id:1) and (id:2) of Definition 4.4 are satisfied.

Proof. Follows directly from the discussion of Definition 4.4 and [20, 2.14]. □

Definition 4.6 (Generalized join dependencies). The constructions of Definition 4.4 characterize completely the conditions needed for a decomposition into two views to be independent relative to a common view, but they do not specify completely the minimal constraints on the main schema $\mathbf{D}$ which enable that decomposition. Consider, once again, the relational schema $R[ABC]$, this time without the FD $B \rightarrow C$, decomposed into the two projection views Π_{AB} and Π_{BC}, let $r_1 = \{(a_1, b_1, c_1), (a_2, b_1, c_2)\}$, and let $r_2 = \{(a_1, b_1, c_1), (a_1, b_1, c_2), (a_2, b_1, c_1), (a_2, b_1, c_2)\}$ be relations on $R[ABC]$. Both r_1 and r_2 map to $(\{(a_1, b_1), (a_2, b_1)\}, \{(b_1, c_1), (b_1, c_2)\})$ under the decomposition into the AB- and BC-projections, but only one may be a legal state of $R[ABC]$ under the constraints of the schema if the decomposition is to be lossless. The classical relational theory mandates that the join dependency $\bowtie [AB, BC]$ hold on $R[ABC]$, so that r_1 is excluded. However, the general theory outlined in Definition 4.4 makes no such preference. There could just as easily be a complex set of constraints on $R[ABC]$ under which r_1 is legal but r_2 is not.

In the general decomposition context developed in [20], it does not appear to be possible to express such a preference in a natural way. However, in the present context, in which database states are expressed as sets of elements, it is quite possible. Specifically, return to the general context of a PF-schema $\mathbf{D} = (\mathsf{SynFnd}(\mathbf{D}), \mathsf{LDB}(\mathbf{D}))$, with a pair $\{\Gamma_1, \Gamma_2\}$ of meet-complementary PF-views whose meet is Γ_3.

(a) The *join completion* of $M \in \mathsf{DB}(\mathbf{D})$ relative to $\{\Gamma_1, \Gamma_2\}$ is

$$\begin{aligned}\mathsf{JoinCompl}_{\langle\Gamma_1;\Gamma_2\rangle}(M) = \{&x \in \mathsf{SynFnd}(\mathbf{D}) \mid \\ &(\exists y \in M)(\exists z \in M)(\gamma^\sharp{}_1(x) = \gamma^\sharp{}_1(y) \wedge \gamma^\sharp{}_2(x) = \gamma^\sharp{}_2(x))\}\end{aligned}$$

(b) Call $M \in \mathsf{DB}(\mathbf{D})$ *join complete* relative to $\{\Gamma_1, \Gamma_2\}$ if $M = \mathsf{JoinCompl}_{\langle\Gamma_1;\Gamma_2\rangle}(M)$. It is easy to see that M is join complete iff the following *generalized join dependency*, denoted $\bowtie [\Gamma_1, \Gamma_2]$, is satisfied.

$$\begin{aligned}\text{(gjd)}\quad &(\forall x, y \in M)((\gamma^\sharp{}_3(x) = \gamma^\sharp{}_3(y)) \Rightarrow \\ &\quad(\exists z \in M)(\gamma^\sharp{}_1(z) = \gamma^\sharp{}_1(x) \wedge (\gamma^\sharp{}_2(z) = \gamma^\sharp{}_2(y))))\end{aligned}$$

(c) Say that $\mathbf{D}$ uses *join reconstruction* from $\{\Gamma_1, \Gamma_2\}$ if every $M \in \mathsf{LDB}(\mathbf{D})$ is join complete relative to $\{\Gamma_1, \Gamma_2\}$.

The theory of this paper will generally be formulated under the explicit stipulation that the decomposition of the main schema $\mathbf{D}$ is governed by the join reconstruct-ion from $\{\Gamma_1, \Gamma_2\}$. In such a context, it is necessary to work with databases which,
while not necessary in $\mathsf{LDB}(\mathbf{D})$, have the property that their join completions are legal. The formal definition is as follows.

(d) Call $M \in \mathsf{DB}(\mathbf{D})$ a *join premodel* relative to $\langle\Gamma_1; \Gamma_2\rangle$ if $\mathsf{JoinCompl}_{\langle\Gamma_1;\Gamma_2\rangle}(M) \in \mathsf{LDB}(\mathbf{D})$. The set of all join premodels of $\mathbf{D}$ relative to $\mathsf{JPair}\langle\Gamma_1; \Gamma 2\rangle$ is denoted $\mathsf{PLDB}\ (\langle\overline{\mathbf{D}}; \{\Gamma_1, \Gamma_2\}\rangle)$.

(e) The schema $\mathbf{D}$ has *implicit join completion* with respect to $\{\Gamma_1, \Gamma_2\}$ if whenever $M \in \mathsf{DB}(\mathbf{D})$ has the property that $\bar{\gamma}_i(M) \in \mathsf{LDB}(\mathbf{V}_i)$ for both $i = 1$ and $i = 2$, then $M \in \mathsf{LDB}(\mathbf{D})$; *i.e.*, $M = \mathsf{JoinCompl}_{\langle\Gamma_1;\Gamma_2\rangle}(M)$.

For example, the schema $\mathbf{E}_2$ of Example 1.2 has implicit join completion, because the join dependency $\bowtie [ABCE, ABCD]$ is implied by the FD $ABC \to E$, and the latter is satisfied by any legal state of the schema of Π_{ABCE}.

Observation 4.7 (Premodels and view axiomatization). Let $\mathbf{D}$ be a PF-schema with $\{\Gamma_1, \Gamma_2\}$ a meet-complementary pair of PF-views of $\mathbf{D}$. Assume further that $\mathbf{D}$ has implicit join completion with respect to $\{\Gamma_1, \Gamma_2\}$. Then for any $M \in \mathsf{DB}(\mathbf{D})$, $M \in \mathsf{PLDB}$ $(\langle\overline{\mathbf{D}}; \{\Gamma_1, \Gamma_2\}\rangle)$ iff $\bar{\gamma}_i(M) \in \mathsf{LDB}(\mathbf{V_i})$ for both $i = 1$ and $i = 2$. □

Definition 4.8 (Join schemata). In order to study the complexity of constraint satisfaction and updates, it is necessary to extend the semantic views; *i.e.*, those views which operate on collections of the form $\mathsf{LDB}(-)$, to syntactic views; *i.e.*, those which operate on collections of the from $\mathsf{DB}(-)$. As identified in Definitions 3.4(h) and 3.7(b), the information needed to effect this translation is built into the PF-view itself. However, there is one important detail which cannot be overlooked; namely, if $\{\Gamma_1, \Gamma_2\}$ is a meet-complementary pair of PF-views on the PF-schema $\mathbf{D}$, it is not generally the case that $\{\overline{\Gamma}_1, \overline{\Gamma}_2\}$ is a meet-complementary pair on $\overline{\mathbf{D}}$. The problem is that $\overline{\gamma}_1^{\flat} \otimes \overline{\gamma}_2^{\flat} : \overline{\mathbf{D}} \to \overline{\mathbf{V}}_1 {}_{\overline{\gamma}_{11}}\!\otimes_{\overline{\gamma}_2} \overline{\mathbf{V}}_2$ is not necessarily injective, although it is certainly surjective. To render it injective, the domain must be restricted to those elements of $\mathsf{DB}(\mathbf{D})$ which are join complete, in the sense of Definition 4.6(b) above.

More formally, let $\mathbf{D} = (\mathsf{SynFnd}(\mathbf{D}), \mathsf{LDB}(\mathbf{D}))$ be a PF-schema, and let $\{\Gamma_1, \Gamma_2\}$ be a meet complementary pair of views of $\mathbf{D}$. Assume further that $\mathbf{D}$ uses join reconstruction from $\{\Gamma_1, \Gamma_2\}$.

(a) Define $\langle\overline{\mathbf{D}}; \{\Gamma_1, \Gamma_2\}\rangle$ to be the schema with $\mathsf{SynFnd}(\langle\overline{\mathbf{D}}; \{\Gamma_1, \Gamma_2\}\rangle) = \mathsf{SynFnd}(\mathbf{D})$ and $\mathsf{LDB}(\langle\overline{\mathbf{D}};\ \{\Gamma_1,\ \Gamma_2\}\rangle) = \{\mathsf{JoinCompl}_{\langle\Gamma_1;\Gamma_2\rangle}(M) \mid M \in \mathsf{DB}(\mathbf{D})\}$. The notation $\mathsf{JDB}_{\langle\Gamma_1,\Gamma_2\rangle}(\overline{\mathbf{D}})$ will often be used to denote $\mathsf{LDB}(\langle\overline{\mathbf{D}}; \{\Gamma_1, \Gamma_2\}\rangle)$.

(b) The view $\widehat{\Gamma}_i = (\overline{\mathbf{V}}_i, \overline{\gamma}_i)$ $(i \in \{1,2\})$ of $\langle\overline{\mathbf{D}}; \{\Gamma_1, \Gamma_2\}\rangle$ is given by $\widehat{\gamma}_i = \bar{\gamma}_{i\,|\,\mathsf{JDB}_{\langle\Gamma_1;\Gamma_2\rangle}(\overline{\mathbf{D}})}$; that is, $\widehat{\gamma}_i$ is just $\bar{\gamma}_i$ restricted to the join-complete members of $\mathsf{DB}(\mathbf{D})$.

It is easy to see that $\widehat{\gamma}_i$ remains surjective, since for any $M \in \mathsf{DB}(\mathbf{D})$, $\bar{\gamma}_i(M) = \bar{\gamma}_i$ $(\mathsf{JoinCompl}_{\langle\Gamma_1,\Gamma_2\rangle}(M))$.

It should also be stressed that the 'hat' notation; *e.g.*, $\widehat{\gamma}_i$, is ambiguous, since the definition depends not only upon γ_i, but upon the meet-complementary pair $\{\Gamma_1, \Gamma_2\}$ as well. However, if this pair is clearly fixed by context, then the meaning of things such as $\widehat{\gamma}_i$ will be apparent and unambiguous.

It is also useful to have some terminology which describes pairs, and sets of pairs, which are compatible with respect to a decomposition.

(c) Call a pair $(M_1, M_2) \in \mathsf{DB}(\mathbf{V}_1) \times \mathsf{DB}(\mathbf{V}_2)$ *join compatible for* $\langle\Gamma_1; \Gamma_2\rangle$ if there is an $M \in \mathsf{JDB}_{\langle\Gamma_1;\Gamma_2\rangle}(\overline{\mathbf{D}})$ such that $(\hat{\gamma}_1^{\flat} \otimes \hat{\gamma}_2^{\flat})(M) = (M_1, M_2)$.

Proposition 4.9. Let $(\mathsf{SynFnd}(\mathbf{D}), \mathsf{LDB}(\mathbf{D}))$ be a PF-schema, and let $\{\Gamma_1, \Gamma_2$ be a meet-complementary pair of PF-views over $\mathbf{D}$. Assume further that $\mathbf{D}$ uses join reconstruction from $\{\Gamma_1, \Gamma_2\}$.

(a) $\{\widehat{\Gamma}_1, \widehat{\Gamma}_2\}$ is a meet complementary pair of PF-views over $\langle\overline{\mathbf{D}}; \{\Gamma_1; \Gamma_2\}\rangle$.

(b) $\hat{\gamma}_1^\flat \otimes \hat{\gamma}_2^\flat = (\overline{\gamma}_1^\flat \otimes \overline{\gamma}_2^\flat)_{|\mathsf{JDB}_{\langle\Gamma_1;\Gamma_2\rangle}(\overline{\mathbf{D}})}$, with the last entry denoting the restriction of $\overline{\gamma}_1 \otimes \overline{\gamma}_2$ to the domain $\mathsf{JDB}_{\langle\Gamma_1;\Gamma_2\rangle}(\overline{\mathbf{D}})$.

Proof. To show (a), first note that since $\mathsf{JDB}_{\langle\Gamma_1;\Gamma_2\rangle}(\overline{\mathbf{D}})$ consists entirely of join complete sets, $\hat{\gamma}_1^\flat \otimes \hat{\gamma}_2^\flat : \mathsf{JDB}_{\langle\Gamma_1;\Gamma_2\rangle}(\overline{\mathbf{D}}) \to \overline{\mathbf{V}}_1 \ {}_{\hat{\gamma}_1}\!\otimes_{\hat{\gamma}_2} \overline{\mathbf{V}}_2$ must be injective. Hence, in view of 4.2, $\{\widehat{\Gamma}_1, \widehat{\Gamma}_2\}$ must be a subdirect complementary pair. To show that it is meet complementary, it suffices to observe that for $M \in \mathsf{JDB}_{\langle\Gamma_1;\Gamma_2\rangle}(\overline{\mathbf{D}})$, letting $M_1 = \widehat{\gamma}_1(M)$ and $M_2 = \widehat{\gamma}_2(M)$, conditions (id:1) and (id:2) of Definition 4.4 are satisfied, and so by propositon 4.5, $\{\widehat{\Gamma}_1, \widehat{\Gamma}_2\}$ is a meet-complementary pair.

Part (b) follows directly from Definition 4.8(c). □

5. Complexity of view constraints in a general setting

In this section, the relative complexity of verifying the constraints on a schema via a meet-complementary pair of its views is investigated. For schemata governed by consistency constraints which do not involve the generation of new elements, the conclusion is strong and positive: in a general sense, a cover of the constraints on the main schema embeds in the view, and so the complexity of constraint checking via the views is no more complex than constraint checking on the main schema itself. This extends the result which was reported in [21], and this is accomplished with a much simpler proof technique.

For constraints which can generate new tuples (such a join dependencies), the results require a condition called *meet uniformity*, which stipulates that the two views comprising the decomposition treat this common sub-view in a uniform way. While this condition is somewhat strict, the constructions nonetheless provide valuable insights.

Notation 5.1. Throughout this section, unless noted specifically to the contrary, $\mathbf{D} = (\mathsf{SynFnd}(\mathbf{D}), \mathsf{LDB}(\mathbf{D}))$ will be taken to be a PF-schema.

Definition 5.2 (The *k*-submodel property). In the earlier work [21], the *k*-submodel property was used as the measure of constraint complexity. To motivate the extended framework developed in this section, it is instructive to provide first a brief summary of this idea.

Given a relation M and a set $\mathcal{F}$ of FDs, it may be determined whether M satisfies those FDs by checking two tuples at a time. If M does not satisfy all of the dependencies in $\mathcal{F}$, then there must be some $\varphi \in \mathcal{F}$ and some two-element subset $N \subseteq M$ with the property that φ does not hold on N. In other words, M satisfies $\mathcal{F}$ iff every subset of M of size two satisfies $\mathcal{F}$. The *k*-submodel property generalizes this idea of reducing constraint satisfaction to checking satisfaction on subinstances of a fixed size.

Formally, for $k \in \mathbb{N}$, a *k-model* of $\mathbf{D}$ is any $M \in \mathsf{DB}(\mathbf{D})$ with the property that for every $N \subseteq M$ with $\mathsf{Card}(N) \leqslant k$, it is necessarily the case that $N \in \mathsf{LDB}(\mathbf{D})$. The schema $\mathbf{D}$ has the *k-submodel property* if for every $M \in \mathsf{DB}(\mathbf{D})$, $M \in \mathsf{LDB}(\mathbf{D})$ iff M is a

k-model of $\mathbf{D}$. If $\mathbf{D}$ has the k-submodel property, then it has the *unrestricted submodel property* as well, in the sense that if $M \in \mathsf{LDB}(\mathbf{D})$, then so too is every subset of M [21, 3.7].

In particular, a relational schema constrained solely by FDs has the 2-submodel property. More generally, the equality-generating constraints, or EGDs, of this classical relational framework, are characterizable via the k-submodel property. See Summary 6.1 and Discussion 6.2 for more details.

Unfortunately, the k-submodel property cannot model any sort of relational constraint which mandates the existence of new tuples based upon existing ones. This includes both full tuple-generating dependencies (TGDs) and embedded TGDs, such as inclusion dependencies, of which the immensely important foreign-key constraints are a special case.

In order to model these more general constraints, an expanded notion of complexity is introduced. Rather than simply asking whether every subset of M of a given size is a model, it asks whether every subset of M of a given size is contained in a model which in turn is contained in M. The formalization of this notion begins with the concept of a completion.

Definition 5.3 (Completions). Let $M \in \mathsf{DB}(\mathbf{D})$. A *completion* of M is any $N \in \mathsf{LDB}(\mathbf{D})$ which contains M. There are two important families of completions of M, which are detailed below.

(a) The *full set of completions* of M is $\mathsf{FullCompl}_{\mathbf{D}}(M) = \{N \in \mathsf{LDB}(\mathbf{D}) \mid M \subseteq N\}$.

(b) The set of *minimal completions* of M is $\mathsf{MinCompl}_{\mathbf{D}}(M) = \{S \in \mathsf{FullCompl}_{\mathbf{D}}(M) \mid (\forall T \in \mathsf{FullCompl}_{\mathbf{D}}(M))((T \subseteq S) \Rightarrow (T = S))\}$. A minimal completion of M is thus a completion which does not contain any proper subset which is itself a completion of M.

Observe that $M \in \mathsf{LDB}(\mathbf{D})$ iff $\mathsf{MinCompl}_{\mathbf{D}}(M) = \{M\}$.

Because a schema with the k-submodel property has the unrestricted submodel property as well, it follows that the only possibilities for the set of minimal completions of $M \in \mathsf{DB}(\mathbf{D})$ in that case are $\{M\}$ (in which case M is already in $\mathsf{LDB}(\mathbf{D})$) and $\emptyset$ (in which case no extension of M is in $\mathsf{LDB}(\mathbf{D})$).

Definition 5.4 ((k_1, k_2)-boundedness). Clearly, the process of completion involves an increase in size of the database. The notion of (k_1, k_2)-boundedness characterizes this growth locally; that is, for models of size no more than k_1. Informally, the (k_1, k_2)-boundedness property states that for any $M \in \mathsf{DB}(\mathbf{D})$ with $\mathsf{Card}(M) \leqslant k_1$, the size of any minimal completion is no more than k_2. It is necessary to be a bit careful here. Even though all databases are taken to be finite, there may be no bound on the size of completions of databases of size k_1. To model this possibility, let $\overline{\mathbb{N}}$ denote the set consisting of the natural numbers, together with a special element which will be denoted by ∞. The (k_1, ∞)-boundedness property then provides no information on the maximum size of completions of databases of size at most k_1. The symbol ∞ will be used rather loosely, at least to the extent that it may be compared to any natural number; *i.e.*, m for any $m \in \mathbb{N}$. The formal definition is as follows.

(a) For $k_1 \in \mathbb{N}$, $k_2 \in \overline{\mathbb{N}}$, the schema $\mathbf{D}$ has the (k_1, k_2)*-boundedness property* if for every $M \in \mathsf{DB}(\mathbf{D})$ with $\mathsf{Card}(M) \leqslant k_1$, every $N \in \mathsf{MinCompl}_{\mathbf{D}}(M)$ has the property that $\mathsf{Card}(N) \leqslant k_2$.

As a specific example, a universal relational schema constrained by FDs (or even EGDs) and a single join dependency has the (2,4)-boundedness property.

Definition 5.5 (The k_1-premodel property). The notion of a k-premodel and the k-premodel property generalize the notions of k-model and the k-submodel property to the context in which completions can add new elements. A k-premodel contains a completion of each of its subsets of size at most k, with a schema having the k-premodel property iff its legal databases are characterized by k-premodels.

More formally, Let $k \in \mathbb{N}$.

(a) A *k-premodel* of $\mathbf{D}$ is an $M \in \mathsf{DB}(\mathbf{D})$ with the property that for every $N \subseteq M$ with $\mathsf{Card}(N) \leqslant k$, there is a $P \in \mathsf{MinCompl}_{\mathbf{D}}(N)$ with $P \subseteq M$.

(b) The schema $\mathbf{D}$ has the *k-premodel property* if for every $M \in \mathsf{DB}(\mathbf{D})$, the condition $M \in \mathsf{LDB}(\mathbf{D})$ holds iff M is a k-premodel of $\mathbf{D}$.

Definition 5.6 (Relative (k_1, k_2)-boundedness and relative k_1- premodels). In the framework of [21], it is always the case that the join dependency underlying the decomposition of the base schema $\mathbf{D}$ into meet-complementary views is generated by the other dependencies, which are in turn embeddable into the component views. In that case, every join-compatible pair of legal view states gives rise to a unique state of the main schema; in the terminology of Definition 4.6(e), the main view has implicit join completion. In the more general framework developed here, this need not be the case. Rather, the join dependency may be specified separately, as elaborated in Definition 4.6. This dependency can force a database in the main schema to be much larger than the sum of the sizes of its components under the decomposition, giving an artificially high measure of how large completions can become. Thus, when looking at some completion N of an $M \in \mathsf{DB}(\mathbf{D})$, it is advantageous to separate the increase in size due to the constraints which embed in the view from that caused by the join dependency governing the decomposition. The key is to ask not how large N can be, but rather how large some $N' \in \mathsf{DB}(\mathbf{D})$ which completes to N upon applying the join dependency can be. The formalization of these ideas, for both the k-submodel and the k-premodel contexts, is as follows. Let $k \in \mathbb{N}$, and assume that $\{\Gamma_1, \Gamma_2\}$ forms a meet-complementary pair of PF-views of $\mathbf{D}$, and that $\mathbf{D}$ uses join reconstruction from $\{\Gamma_1, \Gamma_2\}$ (See Definition 4.6(c)).

(a) $M \in \mathsf{DB}(\mathbf{D})$ is a $\langle\Gamma_1;\Gamma_2\rangle$-*relative k-premodel* of $\mathbf{D}$ if for every $N \subseteq M$ with $\mathsf{Card} \leqslant k$, there is a $P \subseteq M$ with $\mathsf{JoinCompl}_{\langle\Gamma_1;\Gamma_2\rangle}(P) \in \mathsf{MinCompl}_{\mathbf{D}}(N)$.

(b) The schema $\mathbf{D}$ has the $\langle\Gamma_1;\Gamma_2\rangle$-*relative k-premodel property* if for every $M \in \mathsf{DB}(\mathbf{D})$, the condition $M \in \mathsf{PLDB}(\langle\overline{\mathbf{D}}; \{\Gamma_1, \Gamma_2\}\rangle)$ holds iff M is a $\langle\Gamma_1;\Gamma_2\rangle$-relative k-premodel of $\mathbf{D}$.

(a′) $M \in \mathsf{DB}(\mathbf{D})$ is a $\langle\Gamma_1;\Gamma_2\rangle$-*relative k-model* of $\mathbf{D}$ if every $N \subseteq M$ with $\mathsf{Card}(N) \leqslant k$ is in $\mathsf{PLDB}(\langle\overline{\mathbf{D}}; \{\Gamma_1, \Gamma_2\}\rangle)$.

(b′) The schema $\mathbf{D}$ has the $\langle\Gamma_1;\Gamma_2\rangle$-*relative k-submodel property* if for every $M \in \mathsf{DB}(\mathbf{D})$, the condition $M \in \mathsf{PLDB}(\langle\overline{\mathbf{D}}; \{\Gamma_1, \Gamma_2\}\rangle)$ holds iff M is a $\langle\Gamma_1;\Gamma_2\rangle$-relative k-model of $\mathbf{D}$.

(c) For $k_1 \in \mathbb{N}$, $k_2 \in \overline{\mathbb{N}}$, the schema $\mathbf{D}$ has the $\langle\Gamma_1;\Gamma_2\rangle$-*relative* (k_1, k_2)-*boundedness property* if for every $M \in \mathsf{DB}(\mathbf{D})$ with $\mathsf{Card}\, M \leqslant k_1$ and every $N \in \mathsf{Mincompl}_{\mathbf{D}}(M)$, there is a $P \subseteq M$ with $\mathsf{JoinCompl}_{\langle\Gamma_1;\Gamma_2\rangle}(P) = N$ and $\mathsf{Card}(P) \leqslant k_2$.

Definition 5.7 (Rules and their containment semantics). Characterization of the k-premodel and k-submodel properties is facilitated greatly through the use of rules on a schema. Informally, a rule on $\mathbf{D}$ states that if certain elements are present, then so too are others.

(a) A *rule* on $\mathbf{D}$ is a pair $\alpha = (\mathsf{Antc}(\alpha), \mathsf{Cnsq}(\alpha))$ in which $\mathsf{Antc}(\alpha) \in \mathsf{DB}(\mathbf{D})$ and $\mathsf{Cnsq}(\alpha) \subseteq \mathsf{DB}(\mathbf{D})$, with $\mathsf{Antc}(\alpha) \subseteq M$ for each $M \in \mathsf{Cnsq}(\alpha)$. $\mathsf{Antc}(\alpha)$ is called the *antecedent* of α, and $\mathsf{Cnsq}(\alpha)$ its set of *consequents*.

(b) The rule α is called a *denial rule* if $\mathsf{Cnsq}(\alpha) = \emptyset$, and a *generator rule* if $\mathsf{Cnsq}(\alpha) \neq \emptyset$.

Roughly, denial rules correspond to constraints such as EGDs, while generator rules correspond to constraints such as TGDs. See Examples 5.9–5.12 below for examples and more details.

(c) $\mathsf{Rules}(\mathbf{D})$ (resp. $\mathsf{GeneratorRules}(\mathbf{D})$, resp. $\mathsf{DenialRules}(\mathbf{D})$) denotes the set of all rules (resp. generator rules, resp. denial rules) on $\mathbf{D}$.

Informally, the containment semantics of a rule simply state that a structure which contains its antecedent (as a subset) must also contain one of its consequents.

(d) The *containment semantics* of rules are defined as follows.

(i) For $\alpha \in \mathsf{Rules}(\mathbf{D})$ and $M \in \mathsf{DB}(\mathbf{D})$, $M \in \mathsf{Mod}(\alpha)$ iff $\mathsf{Antc}(\alpha) \subseteq M$ implies that there is an $N \in \mathsf{Cnsq}(\alpha)$ with $N \subseteq M$.

(ii) For $\mathcal{S} \subseteq \mathsf{Rules}(\mathbf{D})$ and $M \in \mathsf{DB}(\mathbf{D})$, $M \in \mathsf{Mod}(\mathcal{S})$ iff $M \in \mathsf{Mod}(\alpha)$ for each $\alpha \in \mathcal{S}$.

Since containment semantics will be used exclusively in this paper, the notation $\mathsf{Mod}(-)$ will not result in any ambiguity.

(g) The set of *rule constraints* of $\mathbf{D}$ is $\mathsf{RuleConstr}(\mathbf{D}) = \{\alpha \in \mathsf{Rules}(\mathbf{D}) \mid (\forall M \in \mathsf{LDB}(\mathbf{D}))(M \in \mathsf{Mod}(\alpha))\}$.

Definition 5.8 (The completion rules of a schema). The rules which form the foundation of the results developed here are based upon the notion of minimal completion, as defined in Definition 5.3. They are formalized as follows.

(a) Let $M \in \mathsf{DB}(\mathbf{D})$. The *minimal-completion rule* of M is defined to be $\mathsf{MinComplRule}_{\mathbf{D}}(M) = (M, \mathsf{MinCompl}_{\mathbf{D}}(M))$. In other words, $\mathsf{Antc}(\mathsf{MinComplRule}_{\mathbf{D}}(M)) = M$, and $\mathsf{Cnsq}(\mathsf{MinComplRule}_{\mathbf{D}}(M)) = \mathsf{MinCompl}_{\mathbf{D}}(M)$.

(b) Define $\mathsf{MinComplRules}(\mathbf{D}) = \{\mathsf{MinComplRule}_{\mathbf{D}}(M) \mid M \in \mathsf{DB}(\mathbf{D})\}$.

(c) Let $k \in \mathbb{N}$. Define $\mathsf{MinComplRules}^k(\mathbf{D}) = \{\mathsf{MinComplRule}_{\mathbf{D}}(M) \mid (M \in \mathsf{DB}(\mathbf{D}))\}$.

Although the definition of schema semantics via minimal-completion rules has some similarities to the use of classical constraints, there are a number of key differences. In the next four examples, some of these are highlighted.

Example 5.9 (Rule semantics with functional dependencies). Let $\mathbf{E}_5$ be the relational schema with the single relation $R[ABCD]$, constrained by the FDs $\{A \to B, C \to D\}$. It should first of all be noted that rules, as defined here, do not support quantification. Thus, it not possible to express the following first-order representation of the FD $A \to B$ as a single minimal completion rule.

$$\begin{aligned}&(\forall v_1^A)(\forall v_1^B)(\forall v_2^B)(\forall v_1^C)(\forall v_2^C)(\forall v_1^D)(\forall v_2^D)\\ &\qquad((R(v_1^A, v_1^B, v_1^C, v_1^C) \wedge R(v_1^A, v_2^B, v_2^C, v_2^C) \wedge (v_1^B \neq v_2^B)) \Rightarrow \mathbf{false})\end{aligned}$$

Rather, it is necessary to express each ground instance of the formula (*i.e.*, each instance with the variables bound to specific constants) as a distinct rule. A generic logical ground instance for the above expression is

$$((R(\mathrm{a}_1, \mathrm{b}_1, \mathrm{c}_1, \mathrm{d}_1) \wedge R(\mathrm{a}_1, \mathrm{b}_2, \mathrm{c}_2, \mathrm{d}_2)) \Rightarrow \mathbf{false})$$

with a_1, as well as the b_i's, c_i's, and d_i's, values from the appropriate domains, and $\mathrm{b}_1 \neq \mathrm{b}_2$. The corresponding minimal-completion (denial) rule is

$$\left(\alpha_{A \to B}^{\mathbf{E}_5}\right) \qquad (\{(\mathrm{a}_1, \mathrm{b}_1, \mathrm{c}_1, \mathrm{d}_1), (\mathrm{a}_1, \mathrm{b}_2, \mathrm{c}_2, \mathrm{d}_2)\}, \emptyset)$$

Similarly, the FD $C \to D$ is represented by denial rules of the form

$$\left(\alpha_{C \to D}^{\mathbf{E}_5}\right) \qquad (\{(\mathrm{a}_1, \mathrm{b}_1, \mathrm{c}_1, \mathrm{d}_1), (\mathrm{a}_2, \mathrm{b}_2, \mathrm{c}_1, \mathrm{d}_2)\}, \emptyset)$$

in which $\mathrm{d}_1 \neq \mathrm{d}_2$, but the other elements may nor may not be equal. All other ordered pairs of tuples result in (trivial) normal rules. That is, rules of the form

$$(\{(\mathrm{a}_1,\mathrm{b}_1,\mathrm{c}_1,\mathrm{d}_1),(\mathrm{a}_2,\mathrm{b}_2,\mathrm{c}_2,\mathrm{d}_2)\},\{\{(\mathrm{a}_1,\mathrm{b}_1,\mathrm{c}_1,\mathrm{d}_1),(\mathrm{a}_2,\mathrm{b}_2,\mathrm{c}_2,\mathrm{d}_2)\}\})$$

in which $\mathrm{a}_1 \neq \mathrm{a}_2$ and $\mathrm{c}_1 \neq \mathrm{c}_2$ are always in $\mathsf{RuleConstr}(\mathbf{E}_4)$. However, these not need be included explicitly in a constraint set for this schema; it is sufficient that rules of the form $\alpha^{\mathbf{E}_5}_{A\to B}$ and $\alpha^{\mathbf{E}_5}_{C\to D}$ be included. In general, *identity rules*; that is, rules α for which $\mathsf{Cnsq}(\alpha) = \{\mathsf{Antc}(\alpha)\}$, never need be included in the definition of minimal-completion semantics, since the containment semantics of Definition 5.7(d) takes them to be true by default, unless overridden by other rules.

Example 5.10 (Rule semantics with a join dependency). Next, let $\mathbf{E}_6$ be the relational schema with the single relation $R[ABCD]$, constrained by the FD $A \to B$ and the join dependency $\bowtie[ABC, CD]$. The rule $\alpha^{\mathbf{E}_6}_{A\to B}$, identified above, applies to this schema as well. The join dependency $\bowtie[ABC, CD]$ is represented in $\mathbf{E}_6$ by the set of all rules of the form

$$\begin{aligned}&(\alpha^{\mathbf{E}_6}{}_{\bowtie[ABC,CD]})\\&(\{(\mathrm{a}_1,\mathrm{b}_1,\mathrm{c}_1,\mathrm{d}_1),(\mathrm{a}_2,\mathrm{b}_2,\mathrm{c}_1,\mathrm{d}_2)\},\\&\qquad\{\{(\mathrm{a}_1,\mathrm{b}_1,\mathrm{c}_1,\mathrm{d}_1),(\mathrm{a}_2,\mathrm{b}_2,\mathrm{c}_1,\mathrm{d}_2),(\mathrm{a}_1,\mathrm{b}_1,\mathrm{c}_1,\mathrm{d}_2),(\mathrm{a}_2,\mathrm{b}_2,\mathrm{c}_1,\mathrm{d}_1)\}\})\end{aligned}$$

with either $\mathrm{a}_1 \neq \mathrm{a}_2$ or else $\mathrm{b}_1 = \mathrm{b}_2$. Note that this rule expresses the JD $\bowtie[ABC, CD]$ only in the context of tuples which also satisfy the FD $A \to B$. It is not possible to drop the conditions that either $\mathrm{a}_1 \neq \mathrm{a}_2$ or else $\mathrm{b}_1 = \mathrm{b}_2$. Minimal-completion rules, unlike classical integrity constraints, must have consequents whose members satisfy all of the constraints on the schema. On the other hand, if the FD $A \to B$ on $\mathbf{E}_6$ is dropped, so that the JD is its only constraint, then instances of the above rule with both $\mathrm{a}_1 = \mathrm{a}_2$ and $\mathrm{b}_1 \neq \mathrm{b}_2$ would apply.

Note also that this schema has the $(2, 4)$-boundedness property.

Example 5.11 (Rule semantics with multiple generating dependencies). A third example illustrates this integration of integrity constraints within rules more saliently. Let $\mathbf{E}_7$ have three binary relation symbols $R[AB]$, $S[AB]$, and $T[AB]$, and assume further that $\mathsf{Dom}(A) = \mathsf{Dom}(B)$. Informally, the constraints on $\mathbf{E}_7$ state that S contains R, as well as the composition of R with itself, and T contains S, as well as the composition of S with itself. More formally, these constraints are expressed by the following logical formulas.

$$\begin{aligned}(\forall v_1^A)(\forall v_1^B)(R(v_1^A, v_1^B) &\Rightarrow S(v_1^A, v_1^B))\\(\forall v_1^A)(\forall v_1^B)(S(v_1^A, v_1^B) &\Rightarrow T(v_1^A, v_1^B))\\(\forall v_1^A)(\forall v_2^A)(\forall v_1^B)(\forall v_2^B)(R(v_1^A, v_1^B)\wedge R(v_2^A, v_2^B)\wedge(v_1^B = v_2^A) &\Rightarrow S(v_1^A, v_2^B))\\(\forall v_1^A)(\forall v_2^A)(\forall v_1^B)(\forall v_2^B)(S(v_1^A, v_1^B)\wedge S(v_2^A, v_2^B)\wedge(v_1^B = v_2^A) &\Rightarrow T(v_1^A, v_2^B))\end{aligned}$$

The minimal-completion rules of $\mathbf{E}_7$ must embody all of these constraints simultaneously. For single tuples, the rules take on the following forms. (Since tuples may now come from one of several relations, they must be tagged with the name of the relation of origin.)

$$\begin{array}{ll}
(\alpha_1^{\mathbf{E}_7}) & (\{R(\mathrm{a}_1,\mathrm{b}_1)\},\{\{R(\mathrm{a}_1,\mathrm{b}_1),S(\mathrm{a}_1,\mathrm{b}_1),T(\mathrm{a}_1,\mathrm{b}_1)\}\}) \\
(\alpha_2^{\mathbf{E}_7}) & (\{S(\mathrm{a}_1,\mathrm{b}_1)\},\{\{S(\mathrm{a}_1,\mathrm{b}_1),T(\mathrm{a}_1,\mathrm{b}_1)\}\}) \\
(\alpha_3^{\mathbf{E}_7}) & (\{T(\mathrm{a}_1,\mathrm{b}_1)\},\{\{T(\mathrm{a}_1,\mathrm{b}_1))\}\}
\end{array}$$

The third rule is an identity and will not affect the semantics of the total constraint set. Now, for rules with two elements in the antecedent set, it is best to begin with relation S. There are two possibilities for an antecedent containing two tuples. First, if $\mathrm{a}_1 \neq \mathrm{b}_2$, there are rules of the following form.

$$\begin{array}{ll}
(\alpha_4^{\mathbf{E}_7}) & (\{S(\mathrm{a}_1,\mathrm{b}_1),S(\mathrm{b}_1,\mathrm{b}_2)\}, \\
& \{\{S(\mathrm{a}_1,\mathrm{b}_1),S(\mathrm{b}_1,\mathrm{b}_2),T(\mathrm{a}_1,\mathrm{b}_1),T(\mathrm{b}_1,\mathrm{b}_2),T(\mathrm{a}_1,\mathrm{b}_2)\}\})
\end{array}$$

if $\mathrm{a}_1 = \mathrm{b}_2$, the rules take this form.

$$\begin{array}{ll}
(\alpha_5^{\mathbf{E}_7}) & (\{S(\mathrm{a}_1,\mathrm{b}_1),S(\mathrm{b}_1,\mathrm{a}_1)\}, \\
& \{\{S(\mathrm{a}_1,\mathrm{b}_1),S(\mathrm{b}_1,\mathrm{a}_1),T(\mathrm{a}_1,\mathrm{b}_1),T(\mathrm{b}_1,\mathrm{a}_1),T(\mathrm{a}_1,\mathrm{a}_1),T(\mathrm{b}_1,\mathrm{b}_1)\}\})
\end{array}$$

The rules with two antecedent tuples in R are similar, but more complex, since they involve three relation symbols. First, if $\mathrm{a}_1 \neq \mathrm{b}_2$, the rules have the following form.

$$\begin{array}{ll}
(\alpha_6^{\mathbf{E}_7}) & (\{R(\mathrm{a}_1,\mathrm{b}_1),R(\mathrm{b}_1,\mathrm{b}_2)\}, \\
& \{\{R(\mathrm{a}_1,\mathrm{b}_1),R(\mathrm{b}_1,\mathrm{b}_2),S(\mathrm{a}_1,\mathrm{b}_1),S(\mathrm{b}_1,\mathrm{b}_2),S(\mathrm{a}_1,\mathrm{b}_2), \\
& \qquad T(\mathrm{a}_1,\mathrm{b}_1),T(\mathrm{b}_1,\mathrm{b}_2),T(\mathrm{a}_1,\mathrm{b}_2)\}\})
\end{array}$$

if $\mathrm{a}_1 = \mathrm{b}_2$, the rules take this form.

$$\begin{array}{ll}
(\alpha_7^{\mathbf{E}_7}) & (\{R(\mathrm{a}_1,\mathrm{b}_1),R(\mathrm{b}_1,\mathrm{a}_1)\}, \\
& \{\{R(\mathrm{a}_1,\mathrm{b}_1),R(\mathrm{b}_1,\mathrm{a}_1),S(\mathrm{a}_1,\mathrm{b}_1),S(\mathrm{b}_1,\mathrm{a}_1),S(\mathrm{a}_1,\mathrm{a}_1),S(\mathrm{b}_1,\mathrm{b}_1), \\
& \qquad T(\mathrm{a}_1,\mathrm{b}_1),T(\mathrm{b}_1,\mathrm{a}_1),T(\mathrm{a}_1,\mathrm{a}_1),T(\mathrm{b}_1,\mathrm{b}_1)\}\})
\end{array}$$

The rules of the form $\alpha_1^{\mathbf{E}_7} - \alpha_2^{\mathbf{E}_7}$ and $\alpha_4^{\mathbf{E}_7} - \alpha_7^{\mathbf{E}_7}$ suffice to identify the legal databases of $\mathbf{E}_7$, in the sense that the models of these rules under the containment semantics of Definition 5.7(d) are sufficient to identify $\mathsf{LDB}(\mathbf{E}_7)$. Since these rules all have antecedents of cardinality no more than two, this schema has the two-premodel property, although it does not have the two-submodel property. It also has the $(2, 10)$-boundedness property, but it does not have the $(2, k)$-boundedness property for any

$k < 10$. This is the case even though the logical constraints, as identified above, have a sort of $(2, 3)$-boundedness property, since each takes at most two antecedents and generates at most one additional consequent. Thus, the size of the consequents of a generating rule can be much larger than those of a logical constraint.

Example 5.12 (Rule semantics with an inclusion dependency). Let $\mathbf{E}_8$ consist of two binary relations $R[AB]$ and $S[AB]$, and suppose that it is constrained by the FD $A \rightarrow B$ on S, as well as the inclusion dependency $R[A] \subseteq S[A]$. In other words, $R[A]$ is a foreign key for $S[A]$. The rules which enforce the FD have the following form, for $\mathrm{b}_1 \neq \mathrm{b}_2$.

$$(\alpha_1^{\mathbf{E}_8}) \qquad (\{S(\mathrm{a}_1, \mathrm{b}_1), S(\mathrm{a}_1, \mathrm{b}_2,)\}, \emptyset)$$

The rules which enforce the foreign-key dependency involve a possibly infinite disjunction (depending upon the cardinality of the domain B). They have the following form.

$$(\alpha_2^{\mathbf{E}_8}) \qquad (\{R(\mathrm{a}_1, \mathrm{b}_1)\}, \{\{R(\mathrm{a}_1, \mathrm{b}_1), S(\mathrm{a}_1, b)\} \mid b \in \mathsf{Dom}(B)\})$$

The reason that these examples have been presented is to illustrate the essential difference between minimal-completion rules and ordinary database dependencies. In effect, the right hand side of a minimal-completion rule must always be a collection of *legal* databases, which may render the representation somewhat more complex. As shall be shown, minimal-completion rules are precisely the kind of representation which is necessary for the study of relative complexity of view axiomatization.

Definition 5.13 (Projection of rules to views). To determine the constraints which a rule on the main schema imposes upon a view, that rule is projected onto the view. The definition provided here applies only to minimal-completion rules, but this is not a problem since only such rules are employed in this work. Formally, let $\Gamma = (\mathbf{V}, \gamma)$ be a PF-view of $\mathbf{D}$.

(a) For $\alpha = \mathsf{MinComplRule}_{\mathbf{D}}(M)$, define the *projection of* α *onto* Γ to be the rule $\gamma(\alpha) = \mathsf{MinComplRule}_{\mathbf{V}}(\bar{\gamma}(M))$.

(b) For $\mathcal{S} \subseteq \mathsf{Rules}(\mathbf{D})$, $\gamma(\mathcal{S}) = \{\gamma(\alpha) \mid \alpha \in \mathcal{S}\}$.

(c) The set $\mathcal{S} \subseteq \mathsf{Rules}(\mathbf{D})$ is said to *define* Γ *semantically* if $\mathsf{LDB} = \mathsf{Mod}(\gamma(\mathcal{S}))$.

See the discussion which follows Definition 5.17(c) for an example of a view which is not semantically definable.

Notation 5.14 Further notational conventions For the rest of this section, the focus will be upon the behavior of minimal-completion rules within the context of a meet-complementary pair. To avoid repeating the context over and over, in addition to

$\mathbf{D} = (\mathsf{SynFnd}(\mathbf{D}), \mathsf{LDB}(\mathbf{D}))$ being a PF-schema, unless stated specifically to the contrary, it will be assumed that $\Gamma_1 = (\mathbf{V}_1, \gamma_1)$ and $\Gamma_2 = (\mathbf{V}_2, \gamma_2)$ form a meet complementary pair of views of $\mathbf{D}$, with meet $\Gamma_3 = (\mathbf{V}_3, \gamma_3)$. It will further be assumed that $\mathbf{D}$ uses join reconstruction from $\{\Gamma_1, \Gamma_2\}$.

It is not difficult to see that the property of (k_1, k_2)-boundedness, as well as the property of being a k-premodel or k-model, is inherited by the views from the main schema. The following two results formalize these facts.

Observation 5.15 Let $k_1 \in \mathbb{N}$ and $k_2 \in \overline{\mathbb{N}}$, and assume that $\mathbf{D}$ has the $\langle \Gamma_1; \Gamma_2 \rangle$-relative (k_1, k_2)-boundedness property. Then both $\mathbf{V}_1$ and $\mathbf{V}_2$ have the (k_1, k_2)-boundedness property.

Proof. Let $i \in \{1, 2\}$, let $M \in \mathsf{DB}(\mathbf{V}_i)$ with $\mathsf{Card}(M) \leqslant k_1$, and let $N \in \mathsf{MinCompl}_{\mathbf{V}_i}(M)$. Choose $N' \in \mathsf{LDB}(\mathbf{D})$ with the property that $\bar{\gamma}_i(N') = N$, and choose $M' \subseteq N'$ with the property that $\bar{\gamma}_i(M') = M$ and $\mathsf{Card}(M') = \mathsf{Card}(M)$. Then $N' \in \mathsf{FullCompl}_{\mathbf{D}}(M')$, and so there is an $N'' \in \mathsf{MinCompl}_{\mathbf{D}}(M')$ with $N'' \subseteq N'$. Since $\mathbf{D}$ has the $\langle \Gamma_1; \Gamma_2 \rangle$-relative (k_1, k_2)-boundedness property and $\mathsf{Card}(M') = \mathsf{Card}(M) \leqslant k_1$, there is a $P \subseteq N''$ with $\mathsf{JoinCompl}_{\langle \Gamma_1; \Gamma_2 \rangle}(P) = N''$ and $\mathsf{Card}(P) \leqslant k_2$. Now $\bar{\gamma}_i(P) = \bar{\gamma}_i(N'') \subseteq \bar{\gamma}_i(N') = N$, and since $N \in \mathsf{MinCompl}_{\mathbf{V}_i}(M)$, it follows that $\bar{\gamma}_i(P) = N$. Furthermore, $\mathsf{Card}(N) = \mathsf{Card}(\bar{\gamma}_i(P)) \leqslant \mathsf{Card}(P) \leqslant k_2$, which establishes that $\mathbf{V}_i$ has the (k_1, k_2)-boundedness property. □

Lemma 5.16 (Projection of k-premodels and k-models). Let $k \in \mathbb{N}$.

(a) If M is a $\langle \Gamma_1; \Gamma_2 \rangle$-relative k-premodel of $\mathbf{D}$, then $\bar{\gamma}_i(M)$ is a k-premodel of $\mathbf{V}_i$ for both $i = 1$ and $i = 2$.

(b) If M is a $\langle \Gamma_1; \Gamma_2 \rangle$-relative k-model of $\mathbf{D}$, then $\bar{\gamma}_i(M)$ is a k-model of $\mathbf{V}_i$ for both $i = 1$ and $i = 2$.

(c) $\gamma(\mathsf{MinComplRules}^k(\mathbf{D})) = \mathsf{MinComplRules}^k(\mathbf{V})$.

Proof. Choose $i \in \{1, 2\}$. To show (a), let M be a $\langle \Gamma_1; \Gamma_2 \rangle$-relative k-premodel of $\mathbf{D}$, and let $N' \subseteq \bar{\gamma}_i(M)$ with $\mathsf{Card}(N') \leqslant k$. Choose $N \subseteq M$ with the property that $\bar{\gamma}_i(N) = N'$ and $\mathsf{Card}(N) = \mathsf{Card}(N')$. Since $\mathsf{Card}(N) = \mathsf{Card}(N') \leqslant k$, there is $P \subseteq M$ with $\mathsf{JoinCompl}_{\langle \Gamma_1; \Gamma_2 \rangle}(P) \in \mathsf{MinCompl}_{\mathbf{D}} N$. Since $\bar{\gamma}_i(P) = \bar{\gamma}_i(\mathsf{JoinCompl}_{\langle \Gamma_1; \Gamma_2 \rangle}(P)) \in \mathsf{LDB}(\mathbf{V}_i)$, $\bar{\gamma}_i(P) \in \mathsf{FullCompl}_{\mathbf{V}}(N')$, so there is a $Q \subseteq \bar{\gamma}_i(P)$ with $Q \in \mathsf{MinCompl}_{\mathbf{V}}(N')$. Furthermore, since $P \subseteq M$, $Q \subseteq \bar{\gamma}_i(P) \subseteq \bar{\gamma}_i(M)$, whence $\bar{\gamma}_i(M)$ is a k-model of $\mathbf{V}$.

Part (b) is similar to (a). Just choose $P = N$ and $Q = N'$.

To establish part (c), it suffices to observe that for every $M' \in \mathsf{DB}(\mathbf{V})$, there is an $M \in \mathsf{DB}(\mathbf{D})$ with $\bar{\gamma}(M) = M'$ and $\mathsf{Card}(M) = \mathsf{Card}(M')$. □

Definition 5.17 (Notions of axiomatization for decompositions). To address the question of axiomatization of the views in a meet-complementary decomposition, it is first necessary to be very precise about what is meant by an axiomatization of the main schema $\mathbf{D}$. Formally, let $\mathcal{S} \subseteq \mathsf{MinComplRules}(\mathbf{D})$.

(a) $\mathcal{S}$ is called an $\langle \Gamma_1; \Gamma_2 \rangle$-*embeddable axiomatization* of $\mathbf{D}$ if for every $M \in \mathsf{DB}(\mathbf{D})$, $M \in \mathsf{PLDB}(\langle \overline{\mathbf{D}}; \{\Gamma_1, \Gamma_2\} \rangle)$ iff $\bar{\gamma}_i(M) \in \mathsf{Mod}(\gamma_i(\mathcal{S}))$ for both $i = 1$ and $i = 2$.

(b) $\mathcal{S}$ is called a *complete* $\langle \Gamma_1; \Gamma_2 \rangle$-*axiomatization* of $\mathbf{D}$ if for every $M \in \mathsf{DB}(\mathbf{D})$, $\mathsf{Mod}(\gamma_i(\mathcal{S})) = \mathsf{LDB}(\mathbf{V}_i)$ for both $i = 1$ and $i = 2$.

The schema $\mathbf{E}_2$ of Example 1.2 illustrates the distinction between these two notions. Let $\mathcal{R}_2$ be the set of all minimal-completion rules based upon the FDs in $\mathcal{F}_2$, obtained using the techniques outlined in example 5.9. It is straightforward to show that $\pi_{ABCD}(\mathcal{R}_2)$ represents $\{A \to D, B \to D, CD \to A\}$, and $\pi_{ABCE}(\mathcal{R}_2)$ represents $\{A \to E\}$. Thus, $\mathcal{R}_2$ is an $\{\Pi_{ABCD}, \Pi_{ABCE}\}$-embeddable axiomatization of $\mathbf{E}_2$.

On the other hand, it cannot be a complete $\{\Pi_{ABCD}, \Pi_{ABCE}\}$-axiomatization of $\mathbf{E}_2$. Indeed, this is the whole point of the example $\mathbf{E}_2$. The view Π_{ABCE} cannot be axiomatizable by $\pi_{ABCE}(\mathcal{R}_2)$. If it were, it would have the two-submodel property, which means that it would be axiomatizable by FDs. However, as shown in Appendix A, it is not even finitely axiomatizable.

It is now possible to establish very sharp results about the decomposition of schemata which have the k-submodel property. The decomposition is always $\langle \Gamma_1; \Gamma_2 \rangle$-embeddable, and in particular, to verify that a database of the main schema is legal, it suffices to verify that the decomposed components are each k-models.

Proposition 5.18. (View axiomatization via k-models). Let $k \in \mathbb{N}$ and let $M \in \mathsf{DB}(\mathbf{D})$. Then M is a $\langle \Gamma_1; \Gamma_2 \rangle$-relative k-model of $\mathbf{D}$ iff $\bar{\gamma}_i(M)$ is a k-model of $\mathbf{V}_i$ for both $i = 1$ and $i = 2$.

Proof. Let $M \in \mathsf{DB}(\mathbf{D})$ have the property that $\bar{\gamma}_i(M)$ is a k-model of $\mathbf{V}_i$ for both $i = 1$ and $i = 2$, and let $N \subseteq M$ with $\mathsf{Card}(N) \leqslant k$. Then $\mathsf{Card}(\bar{\gamma}_i(N)) \leqslant k$ for both $i = 1$ and $i = 2$, so $\bar{\gamma}_i(N) \in \mathsf{LDB}(\mathbf{V}_i)$. Thus $(\hat{\gamma}_1^{\flat} \otimes \hat{\gamma}_2^{\flat})^{-1}(\bar{\gamma}_1(N), \bar{\gamma}_2(N)) = \mathsf{JoinCompl}_{\langle \Gamma_1; \Gamma_2 \rangle}(N) \in \mathsf{LDB}\ (\mathbf{D})$, and so $N \in \mathsf{PLDB}(\langle \overline{\mathbf{D}}; \{\Gamma_1, \Gamma_2\} \rangle)$, whence M is a $\langle \Gamma_1; \Gamma_2 \rangle$-relative k-model of $\mathbf{D}$.

The converse follows immediately from Lemma 5.16(b). □

Theorem 5.19. For any $k \in \overline{\mathbb{N}}$, if $\mathbf{D}$ has the $\langle \Gamma_1; \Gamma_2 \rangle$-relative k-submodel property, then $\mathsf{MinComplRules}^k(\mathbf{D})$ is a $\langle \Gamma_1; \Gamma_2 \rangle$-embeddable axiomatization of $\mathbf{D}$.

Proof. The proof follows immediately from Proposition 5.18 and Lemma 5.16(c). □

Discussion 5.20 (Interpretation in the classical framework). At this point, it is helpful to step back and interpret the above result within the context of the simple example of $\mathbf{E}_2$ presented in example 1.2. Since $\mathbf{E}_2$ is constrained by FDs, it has the two-submodel property. The above proposition states that to determine whether a relation $r[ABCDE]$ satisfies the set $\mathcal{F}_2$ of FDs, it suffices to check whether each subset of $r[ABCD]$ containing at most two tuples and each subset of $r[ABCD]$ containing at most two tuples satisfies the constraints of the view. However, since the projection of a family of FDs is a family of EGDs ([11, Thm. 6.1]), and since the FDs are precisely the EGDs of degree two (see summary 6.1) it follows that this reduces to checking the FDs on Π_{ABCE} and Π_{ABCD}. The important point to note is that this characterization does not require that the projected constraints completely axiomatize each view individually. This has already been observed in the discussion of Definition 5.17. The above result shows that the projected constraints, taken together, nonetheless suffice to characterize those of the main schema $\mathbf{E}_2$.

In the above example, the whole argument may seem rather trivial, since each FD of $\mathcal{F}_2$ embeds into one of the views. However, this need not be the case in general. To illustrate, let $\mathbf{E}_9$ be the schema with the single relation $R[ABC]$, governed by the FDs $\mathcal{F}_9 = \{A \rightarrow B,\ A \rightarrow C,\ B \rightarrow A,\ C \rightarrow A\}$, and consider the pair $\{\Pi_{AB}, \Pi_{BC}\}$ of views. Note that $\mathcal{F}_9$ does not embed into the views. While $A \rightarrow B$ and $B \rightarrow A$ embed into Π_{AB}, the FDs $A \rightarrow C$ and $C \rightarrow A$ embed into neither view, nor are they implied by the pair $\{A \rightarrow B,\ B \rightarrow A\}$. Therefore, a nave approach which considers only those FDs which embed into the views will not deliver the required constraints. Nonetheless, the decomposition $\{\Pi_{AB}, \Pi_{BC}\}$ is easily seen to be meet complementary. In the classical theory, the way to prove this is to find a *cover* of $\mathcal{F}_9$ (*i.e.*, a set of equivalent FDs) which does embed into the two views. One such cover is $\mathcal{F}'_9 = \{A \rightarrow B,\ B \rightarrow C,\ B \rightarrow A,\ C \rightarrow B\}$. which is easily seen to be equivalent to $\mathcal{F}_9$. The set $\{A \rightarrow B, B \rightarrow A\}$ embeds into Π_{AB}, while $\{B \rightarrow C,\ C \rightarrow B\}$ embeds into Π_{BC}.

At first glance, this might seem to contradict the above results. However, it does not. The reason is, once again, the distinction between classical logical constraints and minimal-completion rules. Consider a generic ground rule for the FD $A \rightarrow C$, with $\mathrm{c}_1 \neq \mathrm{c}_2$.

$$(\alpha^{\mathbf{E}_9}_{A\rightarrow C}) \qquad (\{(\mathrm{a}_1, \mathrm{b}_1, \mathrm{c}_1), (\mathrm{a}_1, \mathrm{b}_2, \mathrm{c}_2)\}, \emptyset)$$

If $A \rightarrow C$ were the only constraint in $\mathcal{F}_9$, then both projections of this rule (onto Π_{AB} and onto Π_{BC}) would be identity rules. However, rules must take into consideration the entire constraint set, so in the context of $\mathcal{F}_9$, the projection onto Π_{AB} is the denial $(\{(\mathrm{a}_1, \mathrm{b}_1), (\mathrm{a}_1, \mathrm{b}_2)\}, \emptyset)$ in the case that $\mathrm{b}_1 \neq \mathrm{b}_2$. On the other hand, if $\mathrm{b}_1 = \mathrm{b}_2$, then the projection onto Π_{BC} is the denial $(\{(\mathrm{b}_1, \mathrm{c}_1), (\mathrm{b}_1, \mathrm{c}_2)\}, \emptyset)$, in view of the fact that the FD $B \rightarrow C$ is implied by $\mathcal{F}_9$. Thus, while the rule $\alpha^{\mathbf{E}_9}_{A \rightarrow C}$ above represents only $A \rightarrow C$, the semantics of projection defined in Definition 5.13(b) ensure that the information about the other constraints which must embed into the views is taken into account automatically. The key point of the above result, in the context of denial rules, is that taking this additional information into account does not increase the complexity of the rules. For sets of FDs, this can of course be proved in a more direct fashion, since an embeddable cover of a set of FDs is always as set of FDs. However, for more complex families of constraints, this may not be the case.

Example 5.21 (Difficulties in extension to generating rules). Based upon the strong result expressed in theorem 5.19 for schemata constrained by denial rules, it is natural to conjecture that a similar result holds in the presence of generating rules. Unfortunately, this is not the case. Indeed, for any $n \in \mathbb{N}$, there is a schema $\mathbf{E}_{10}$ with the 2-premodel property, and a meet complementary pair $\{\Omega_{101}, \Omega_{102}\}$ of views of that schema with the property that the schema of Ω_{102} does not have the k-premodel property for any $k \leqslant 2^{n-1}$. The example which illustrates this is quite simple. Let $n \in \mathbb{N}$, and let $A = \{a_i \mid 1 \leqslant i \leqslant 2^n - 1\}$ be an indexed set of elements. Let b be any element not in A, and define $S = A \cup \{b\}$. Regard the elements of A as forming a complete binary tree, represented in sequential fashion [22, Section 2.2.2]. For each vertex a_i with $1 \leqslant \ i \leqslant 2^{n-1}$, the left child is a_{2i} and the right child is a_{2i+1}. Conversely, the parent of a_i for $i > 1$ is $a_{\lfloor i/2 \rfloor}$, with $\lfloor i/2 \rfloor$ denoting $n/2$ rounded down to the nearest integer. The elements of S form the syntactic basis for the PF-schema $\mathbf{E}_{10}$; the elements of $\mathsf{LDB}\ (\mathbf{E}_{10})$ are defined to be precisely those subsets of S which satisfy the following constraints.

(i) A parent vertex of the tree defined by A is in $M \in \mathsf{LDB}(\mathbf{E}_{10})$ iff both of its children are. (Note that one child may be present without the parent.)

(ii) If $a_1 \in M$, then $b \in M$ as well.

It is easy to see that $\mathbf{E}_{10}$ has the two-premodel property.

Now, define two views of $\mathbf{E}_{10}$. The view Ω_{101} has A as the syntactic basis of its schema $\mathbf{E}_{101}$, with the view mapping $M \mapsto M \cap A$. In other words, Ω_{101} preserves the tree of A but drops b. For the view Ω_{102}, define $A' = \{a_i \mid 2^{n-1} \leqslant i \leqslant 2^n - 1\}$, and let the syntactic basis of the underlying schema $\mathbf{E}_{102}$ be $A' \cup \{b\}$, with the view mapping $M \mapsto M \cap (A' \cup \{b\})$. In other words, Ω_{102} preserves b as well as the leaf nodes of the tree formed by A, but drops all interior nodes of that tree. It is easy to see that $\{\Omega_{101}, \Omega_{102}\}$ forms a meet-complementary pair. The meet is the view which keeps just A'; i.e., the leaf nodes of the tree. It follows that for any $M \in \mathsf{LDB}(\mathbf{E}_{10})$, $a_1 \in M$ iff $A' \subseteq M$. In light of the constraint identified in (ii) check the rest above, the following constraint must hold in $\mathbf{E}_{10}$ and so in $\mathbf{E}_{102}$ as well.

(iii) If every element of A' is in $M \in \mathsf{LDB}(\mathbf{E}_{102})$, then so too is b.

However, this constraint cannot be represented any more succinctly within that view, since the structure of the interior vertices of the tree is not available. There are no other constraints on that view. Clearly, this prevents $\mathbf{E}_{102}$ from having the k-premodel property for any $k < 2^{n-1}$.

There is a class of decompositions for which such problems do not occur. These views have the property that if two view states match on the meet, then so too do all completions.

Definition 5.22 (Meet uniformity of rules).

(a) A set $\mathcal{S} \subseteq \mathsf{Rules}(\mathbf{D})$ is called Γ_3-*uniform*, or just *meet uniform*, if whenever $\alpha_1, \alpha_2 \in \mathcal{S} \cap \mathsf{GeneratorRules}(\mathbf{D})$ with $\bar{\gamma}_3(\mathsf{Antc}(\alpha_1)) = \bar{\gamma}_3(\mathsf{Antc}(\alpha_2))$, then for every $M_1 \in \mathsf{Cnsq}(\alpha_1)$ and $M_2 \in \mathsf{Cnsq}(\alpha_2)$, $\bar{\gamma}_3(M_1) = \bar{\gamma}_3(M_2)$.

(b) For $k \in \mathbb{N}$, the schema $\mathbf{D}$ is called $\langle k, \Gamma_3\rangle$-*uniform* if the set $\mathsf{MinComplRules}^k(\mathbf{D})$ is Γ_3-uniform.

Before moving on to the main decomposition result, it is necessary to establish a bound on how large a 'basis' or 'skeleton' for a join-complete set must be, in terms of the size of its projections to the views of the decomposition.

Lemma 5.23 (Representation of join-complete sets). Let $M \in \mathsf{JDB}_{\langle\Gamma_1;\Gamma_2\rangle}(\overline{\mathbf{D}})$. Then there is a set $N \subseteq M$ containing at most $\mathsf{Card}(\bar{\gamma}_1(M)) + \mathsf{Card}(\bar{\gamma}_2(M))$ elements with the property that $\mathsf{JoinCompl}_{\langle\Gamma_1;\Gamma_2\rangle}(N) = M$.

Proof. Begin by partitioning M into three disjoint sets. Let $S_1 = \{a \in M \mid \gamma^\sharp{}_2(a) = \eta_{\mathbf{V}_2}\}$, $S_2 = \{a \in M \mid \gamma^\sharp{}_1(a) = \eta_{\mathbf{V}_1}\}$, and $S_3 = \{a \in M \mid (\gamma^\sharp{}_1(a) \in \mathsf{SynFnd}\ (\mathbf{V}_1))$ *and* $(\gamma^\sharp{}_2(a) \in \mathsf{SynFnd}(\mathbf{V}_2))\}$. (Recall the definition of $\eta_{\mathbf{V}_1}$ from Definition 3.2(c)). In other words, S_1 consists of those elements which project only onto Γ_1, S_2 those which project only onto Γ_2, and S_3 those which project onto both Γ_1 and Γ_2. Next, choose $T_1 \subseteq S_1$ such that $\gamma^\sharp{}_1$ is injective on T_1; that is, $a_1, a_2 \in T_1$ and $\gamma^\sharp{}_1(a_1) = \gamma^\sharp{}_1(a_2)$ implies $a_1 = a_2$. Similarly, choose $T_2 \subseteq S_2$ such that $\gamma^\sharp{}_2$ is injective on T_2. Thus, $\mathsf{Card}(T_i) = \mathsf{Card}(\bar{\gamma}_i(S_i))$ for both $i = 1$ and $i = 2$. For any $Q \subseteq S_3$, call an element $a \in Q$ *redundant for* Q if there are $a_1, a_2 \in Q$ with $a_1 \neq a$, $a_2 \neq a$, and with the property that $\gamma^\sharp{}_i(a_i) = \gamma^\sharp{}_i(a)$ for both $i = 1$ and $i = 2$. Let $T_3 \subseteq S_3$ be obtained from S_3 by repeatedly removing redundant elements, until there are no more. Clearly $\bar{\gamma}_i(T_3) = \bar{\gamma}_i(S_3)$ for both $i = 1$ and $i = 2$. Furthermore, since T_3 does not contain any redundant elements, $\mathsf{Card}(T_3) \leqslant \mathsf{Card}(\bar{\gamma}_1(S_3)) + \mathsf{Card}(\bar{\gamma}_2(S_3))$. Define $N = T_1 \cup T_2 \cup T_3$. Then $\mathsf{Card}(N) = \mathsf{Card}(T_1) + \mathsf{Card}(T_2) + \mathsf{Card}(T_3) \leqslant \mathsf{Card}(\bar{\gamma}_1(S_1)) + \mathsf{Card}(\bar{\gamma}_2(S_2)) + \mathsf{Card}(\bar{\gamma}_1(S_3)) + \mathsf{Card}(\bar{\gamma}_2(S_3)) \leqslant \mathsf{Card}(\bar{\gamma}_1(M)) + \mathsf{Card}(\bar{\gamma}_2(M))$. Furthermore, for both $i = 1$ and $i = 2$, $\bar{\gamma}_i(N) = \bar{\gamma}_i(M)$, so $\mathsf{JoinCompl}_{\langle\Gamma_1;\Gamma_2\rangle}(N) = M$, completing the proof. □

Now, the results of Proposition 5.18 and Theorem 5.19 may be extended to the case of meet-uniform rules.

Proposition 5.24 (View axiomatization via $\langle\Gamma_1;\Gamma_2\rangle$-models). Let $k_1 \in \mathbb{N}$, $k_{12}, k_{22} \in \overline{\mathbb{N}}$, and assume that $\mathbf{D}$ is $\langle k_1, \Gamma_3\rangle$-uniform.

(a) For any $M \in \mathsf{DB}(\mathbf{D})$, M is a $\langle\Gamma_1;\Gamma_2\rangle$-relative k_1-premodel of $\mathbf{D}$ iff $\bar{\gamma}_i(M)$ is a k_1-premodel of $\mathbf{V}_i$ for both $i = 1$ and $i = 2$.

(b) If $\mathbf{V}_1$ has the (k_1, k_{21})-boundedness property, and $\mathbf{V}_2$ has the (k_1, k_{22})-boundedness property, then $\mathbf{D}$ has the $\langle\Gamma_1;\Gamma_2\rangle$-relative $(k_1, k_{21} + k_{22})$-boundedness property.

Proof. To show part (a), let $M \in \mathsf{DB}(\mathbf{D})$ have the property that $\bar{\gamma}_1(M)$ is a k_1-premodel of $\mathbf{V}_1$ and $\bar{\gamma}_2(M)$ is a k-premodel of $\mathbf{V}_2$, and let $N \subseteq M$ with $\mathsf{Card}(N) \leqslant k_1$. Then for both $i = 1$ and $i = 2$, $\mathsf{Card}(\bar{\gamma}_i(N)) \leqslant k_1$ as well, so using the fact that $\bar{\gamma}_i(M)$ is a k_1-premodel, choose $P_i \in \mathsf{MinCompl}_{\mathbf{V}_1}(\bar{\gamma}_i(N))$ with $P_i \subseteq \bar{\gamma}_i(M)$ for $i = 1$ and $i = 2$. Since $\mathbf{D}$ is Γ_3-uniform, it follows that $\lambda\langle\Gamma_1,\Gamma_3\rangle(P_1) = \lambda\langle\Gamma_2,\Gamma_3\rangle(P_2)$, so $(\bar{\gamma}_1 \otimes \bar{\gamma}_2)^{-1}(P_1, P_2) \in \mathsf{LDB}(\mathbf{D})$. For notational convenience, define $P = (\bar{\gamma}_1 \otimes \bar{\gamma}_2)^{-1}(P_1, P_2)$. Since $P_i \subseteq \bar{\gamma}_i(M)$ for both $i = 1$ and $i = 2$, it follows that $P \subseteq \mathsf{JoinCompl}_{\langle\Gamma_1;\Gamma_2\rangle}(M)$. Hence, $P \in \mathsf{FullCompl}_{\mathbf{D}}(N)$. It is furthermore the case that $P \in \mathsf{MinCompl}_{\mathbf{D}}(N)$. Indeed, if $P' \in \mathsf{MinCompl}_{\mathbf{D}}(N)$ with $P' \subseteq P$, then $\bar{\gamma}_i(N) \subseteq \bar{\gamma}_i(P') \subseteq \bar{\gamma}_i(P)$ for both $i = 1$ and $i = 2$, and since $\bar{\gamma}_i(P) = P_i \in \mathsf{MinCompl}_{\mathbf{V}_i}(\bar{\gamma}_i(N))$, it follows that $\bar{\gamma}_i(P') = \bar{\gamma}_i(P) = P_i$ for both $i = 1$ and $i = 2$, whence $P = P'$. Finally, set $P'' = M \cap P$ to obtain $P'' \subseteq M$ with $\mathsf{JoinCompl}_{\langle\Gamma_1;\Gamma_2\rangle}(P'') \in \mathsf{MinCompl}_{\mathbf{D}}(N)$. The converse follows immediately from Lemma 5.16(a).

To show part (b), first note that by assumption, $\mathsf{Card}(P_i) \leqslant k_{2i}$ for $i = 1$ and $i = 2$. Then, rather than choosing $P'' = M \cap P$, use Lemma 5.23 to construct a P'' with $\mathsf{Card}(P'') \leqslant k_{21} + k_{22}$ and $\mathsf{JoinCompl}_{\langle\Gamma_1;\Gamma_2\rangle}(P'') = P$. □

Theorem 5.25. For any $k \in \overline{\mathbb{N}}$, if $\mathbf{D}$ is Γ_3-uniform and has the $\langle\Gamma_1;\Gamma_2\rangle$-relative k-premodel property, then $\mathsf{MinComplRules}^k(\mathbf{D})$ is a $\langle\Gamma_1;\Gamma_2\rangle$-embeddable axiomatization of $\mathbf{D}$.

Proof. The proof follows immediately from Proposition 5.24 and Lemma 5.16(a)+(c). □

Discussion 5.26 (The applicability of meet uniformity). The results of Proposition 5.24 and Theorem 5.25 provide an extension of those of Proposition 5.18 and Theorem 5.19 in a context of relatively strong independence of the two views comprising the meet-complementary pair. Unfortunately, it fails to supports a level of independence which is adequate in some practical situations. For example, let $\mathbf{E}_{11}$ be the relational schema constrained by the single relation $R[ABCDE]$, together with the JD $\bowtie [ABC, CDE]$ and the two inclusion dependencies $R[A] \subseteq R[B]$ and $R[D] \subseteq R[E]$. It is easy to see that the pair $\{\Pi_{ABC}, \Pi_{CDE}\}$ forms a meet-complementary pair, since $R[A] \subseteq R[B]$ embeds in Π_{ABC} and $R[D] \subseteq R[E]$ embeds in Π_{CDE}. Yet, this decomposition is not meet uniform, since the two inclusion dependencies may generate essentially arbitrary values for attribute C.

As shown in example 5.21, in the presence of generating rules, it is nonetheless necessary to exercise some control over the nature of the meet. What seems to be necessary is a way to distinguish the inconsequential additions to it made by the inclusions dependencies of $\mathbf{E}_{11}$ from the clever encoding of information which is illustrated by the example of example 5.21. The development of a theory which addresses this distinction in a general way must be left as a topic for future research. However, as a first step in this direction, in the next section, it is shown that for situations involving the relational model which include $\mathbf{E}_{11}$, it is possible to characterize, in a very useful way, characteristics which a meet-complementary decomposition imposes upon the constraints of the schema.

The results of Proposition 5.18 and Theorem 5.19, as well as Proposition 5.24 and Theorem 5.25, extend those of [21] in a significant way. In that paper, the focus was on the complexity of verifying the correctness of updates on a view. So rather than identifying the complexity of an embedded cover of the constraints of the main schema, only the complexity of verifying that a proposed state of the view schema is correct, under the assumption that the state of the complement is already correct, was established (as 4.14 of that paper). Knowledge of the complexity of the embedded cover provides more information. These ideas are formalized in the following, for both the *k*-submodel and *k*-premodel contexts.

Definition 5.27 (The relative generalized submodel property). Suppose that Γ_1 is to be updated with constant-complement Γ_2. According to [20, 3.10] (see also the summary in Summary 2.1 of this paper), the allowable updates are precisely those which hold the meet $\Gamma_1 \wedge \Gamma_2 = \Gamma_3$ constant. Therefore, when performing such an update, it is known that for any proposed new state $M \in \mathsf{DB}(\mathbf{V}_1)$, its projection onto Γ_3 is already legal; i.e., $\overline{\lambda\langle \Gamma_1, \Gamma_2 \rangle}(M) \in \mathsf{LDB}(\mathbf{V}_3)$. This information may be used to reduce substantially the number constraints which need to be checked; the following definitions formalize these concepts.

Let $k \in \mathbb{N}$ and let $i \in \{1, 2\}$. The ideas below generalize the those of [21, 4.13] for *k*-premodels and *k*-submodels.

(a) The database $M \in \mathsf{DB}(\mathbf{V}_i)$ is called Γ_3-*legal* if $\overline{\lambda\langle \Gamma_i, \Gamma_3 \rangle}(M) \in \mathsf{LDB}(\mathbf{V}_3)$, and it is called a Γ_3-*relative k-premodel* (resp. Γ_3-*relative k-submodel*) for $\mathbf{V}_i$ if it is both Γ_3-legal and a *k*-premodel (resp. *k*-model) of $\mathbf{V}_i$.

(b) The view $\Gamma_i = (\mathbf{V}_i, \gamma_i)$ has the Γ_3-*relative k-premodel property* (resp. Γ_3-*relative k-submodel property*) if, for every $M \in \mathsf{DB}(\mathbf{V}_i)$, the condition $M \in \mathsf{LDB}(\mathbf{V}_i)$ holds iff M is a generalized Γ_3-relative *k*-premodel (resp. a generalized Γ_3-relative *k*-model) for $\mathbf{V}_i$.

The next and final theorem of this section establishes formally that the complexity of testing updates on Γ_1, with Γ_2 held constant, is defined by the generalized Γ_3-relative *k*-submodel property on $\mathbf{V}_1$, even in the case that the full schema $\mathbf{V}_2$ itself has a much higher degree of complexity for its set of constraints.

Theorem 5.28 (Relative complexity for view updates under closed strategies). Let $k \in \mathbb{N}$.

(a) If $\mathbf{D}$ has the $\langle \Gamma_1; \Gamma_2 \rangle$-relative k-submodel property, then both Γ_1 and Γ_2 have the Γ_3-relative k-submodel property.

(b) If $\mathbf{D}$ is Γ_3 uniform and has the $\langle \Gamma_1; \Gamma_2 \rangle$-relative k-premodel property, then both Γ_1 and Γ_2 have the Γ_3-relative k-premodel property.

Proof. The proof follows directly from Theorem 5.19 and Theorem 5.25. □

6. An application within the relational framework

In the classical relational theory – as well as in practice – the two most important types of dependencies are the functional dependencies and the inclusion dependencies. The latter are central to the modelling of *foreign-key constraints*, which are a part of standard SQL and virtually all modern database-management systems. While the theory of the previous section provides very strong results for schemata constrained by functional dependencies (since they satisfy the two-submodel property), it does not provide any results for inclusion dependencies, since they are neither universal (and so do not satisfy the k-submodel property for any k), nor do they define meet-uniform minimal completions (and so Theorem 5.25 does not include schemata which are constrained by them).

Given the importance of these families of constraints, it is essential for any theory of decomposition to address schemata and decompositions in which they occur. In this section, the question of the complexity of view axiomatization is addressed for multi-relation schemata constrained by EGDs and an important subset of the implicational dependencies called *fanout-free unary inclusion dependencies* (fanout-free UINDs). This class of inclusion dependencies is nonetheless quite powerful; in particular, it is sufficient to model foreign-key constraints. Using ideas reported in [7], which show that EGDs and UINDs essentially decouple from one another in terms of inference, it is established that in the context of a meet-complementary decomposition defined by projections of the component relations, if the main schema is constrained by a combination of EGDs of degree at most k and fanout-free UINDs, then to verify that a candidate database of the main schema is legal, it suffices to check that the decomposed components each satisfy all embedded EGDs of degree at most k, as well as all embedded UINDs. In particular, if the EGDs are FDs, then it suffices to check that the both components of the decomposition satisfy all embedded FDs as well as all embedded UINDs.

Summary 6.1 (EGDs and UINDs). The topic of dependencies on relational databases has been studied extensively, if not exhaustively [27]. Therefore, the discussion here is limited to establishing notation and nonstandard conventions.

It is assumed that there is a finite set $\mathbf{U}$ of attributes, as well as a finite set of relation symbols $\{R_i \mid 1 \leqslant i \leqslant n\}$. With each relation symbol is associated a $\mathbf{U}_i \subseteq \mathbf{U}$, with $\mathbf{U}_i \cap \mathbf{U}_j = \emptyset$ for $i \neq j$ and $\cup_{i=1}^{n} \mathbf{U}_i = \mathbf{U}$. There is also an infinite set $\mathsf{Dom}(\mathbf{U})$ called the *universe of domain values*. Following standard conventions, a *tuple* over R_i is a function $t : \mathbf{U}_i \to \mathsf{Dom}(\mathbf{U})$, and a *relation* for R_i is a finite set of tuples over R_i. A *database* is just a set of relations, one for each R_i. Sets of attributes are often written linearly, so $R[ABC]$ is shorthand for $R[\{A, B, C\}]$.

The universal dependencies (*i.e.*, those dependencies which are representable using logical formulas involving only universal quantifiers) which are used in this work are called *full implicational dependencies (FIDs)* [7], or sometimes just *implicational dependencies (IDs)* [11, 23]. These include the functional dependencies (FDs) and join dependencies (JDs) which are ubiquitous in the relational theory. The three references just cited all give readable summaries of these constraints, so there is no need to repeat them here. Suffice to say that they are always *unirelational* (that is, they apply to only one relation), and they are always *typed*; that is, they can make comparisons of tuples only between entries in the same column. Because they are used fundamentally in the results developed in this section, it is perhaps appropriate to say a bit more about the *equality-generating dependencies (EGDs)* [26, Section 3.6], [1, 10.1]. Such dependencies take the following general form:

$$(\forall.))((t_1 \wedge t_2 \wedge \ldots \wedge t_n) \Rightarrow \varepsilon)$$

The terms on the left-hand side are (constant-free) atoms in the language of the relations, while the right-hand side is an equality of two of the variables from the left-hand side. As an example, here is the formula for the functional dependency $A \to B$ on $R[ABC]$.

$$R(x_1, y_1, z_1) \wedge R(x_1, y_2, z_2) \Rightarrow (y_1 = y_2)$$

The *degree* of an EGD is the number of atoms which occur on the left-hand side of the defining formula. The FDs are precisely the EGDs which can be expressed with degree two, and so a relational schema constrained by EGDs has the two-submodel property iff it has a basis consisting of FDs. Although EGDs which are not FDs exist and are not difficult to construct, it is not clear that they have any practical use. Nonetheless, since the theory developed here supports them with no additional effort, and since the inclusion of EGDs in fact shows more clearly how the complexity patterns behave, they are retained in their full generality.

To follow the constructions below (particularly in Lemma 6.16 and Lemma 6.19), it is essential to understand in detail what is meant by a typed constraint. By way of example, the following constraint is not typed, because it compares values in two different columns,

$$R(x_1, y_1, z_1) \wedge R(y_2, x_1, z_2) \Rightarrow (y_1 = y_2)$$

and the following is not typed because it equates values in two different columns.

$$R(x_1, y_1, z_1) \wedge R(x_1, y_2, z_2) \Rightarrow (y_1 = z_2)$$

The family of *inclusion dependencies (INDs)* is neither typed nor universal. In its most general form, for $\mathbf{Y}_i \subseteq \mathbf{U}_i$ and $\mathbf{Y}_j \subseteq \mathbf{U}_j$, the inclusion dependency $R_i[\mathbf{Y}_i] \subseteq R_j[\mathbf{Y}_j]$ stipulates that for every tuple r in the instance of R_i, there is a tuple s in the instance of R_j for which $r[\mathbf{Y}_i] = s[\mathbf{Y}_j]$. In their general form, INDs are very complex. For example, it is known that the inference problem for INDs and FDs together is undecidable [6]. For this reason, various subfamilies of the INDs have been studied as well. For this work, the most important is the *unary INDs (UINDs)*, in which both $\mathbf{Y}_i$ and $\mathbf{Y}_j$ are restricted to contain just one attribute. For UINDs, various associated inference problems are not only decidable, but of reasonable complexity as well. Furthermore, and central to this paper, in the associated inference problem, the effects of the INDs and other dependencies (IDs, etc.) can be essentially 'decoupled' [7]. Since each attribute occurs in exactly one relation, the UID $R_i[\mathbf{Y}_i] \subseteq R_j[\mathbf{Y}_j]$ may be unambiguously abbreviated to just $\mathbf{Y}_i \subseteq \mathbf{Y}_j$. However, since the symbol '$\subseteq$' is already overloaded mathematically, the alternate notation $\mathbf{Y}_i \sqsubseteq \mathbf{Y}_j$ using the squared inclusion symbol will be used in the presentation which follows. (More precisely, since $\mathbf{Y}_i = \{A_i\}$ and $\mathbf{Y}_j = \{A_j\}$ are singletons in the case of UINDs, the resulting notation is just $A_i \sqsubseteq A_j$.) Occasionally, the notation $A_j \sqsupseteq A_i$ will be used; it has the same meaning as $A_i \sqsubseteq A_j$. Similarly, $A_i \sqsubseteq A_j \sqsubseteq A_k$ means that both $A_i \sqsubseteq A_j$ and $A_j \sqsubseteq A_k$ hold.

There is one further point of framework to be made. In much of the classical theory of dependencies, it is further assumed that $\mathsf{Dom}(\mathbf{U})$ is partitioned into disjoint sets $\{\mathsf{Dom}(A) \mid A \in \mathbf{U}\}$, with a tuple entry for attribute A restricted to take values from $\mathsf{Dom}(A)$. In the context of typed dependencies, this makes complete sense. However, since inclusion dependencies are not typed, this convention cannot be used here. Rather, the same domain value must be permitted in different columns, both of the same relation and of different relations. All that can be said is that $\mathsf{Dom}(A) \subseteq \mathsf{Dom}(\mathbf{U})$ for each $A \in \mathbf{U}$.

Finally, it is appropriate to recall a point of notation. If Φ is a set of constraints of some type T (*e.g.*, FIDs, UINDs, EGDs, FDs, etc.), then Φ^+ is used to denote the set of all constraints of that same class which are consequences of those in Φ. For the purposes of this paper, this will always mean *finite* implication. See Lemma 6.7 below for more details.

Discussion 6.2 (Projections of EGDs). The theory developed here is based upon views of relational schemata which are defined by projections of relations onto subsets of their attributes. The problem of characterizing the dependencies on a view is called the *implied constraint problem* [24], of which there are two dimensions. On the one hand, there is the aspect of complexity; as illustrated by the example of example 1.1, the constraints on the view may be far more complex than those on the main schema. On the other hand, there is the aspect of form. Again, the example of Example 1.1 illustrates this, showing that the projection of a family of FDs need not be characterized by FDs.

The main result on form which is needed in this paper states that the projection of a family of EGDs is always characterized by a family of EGDs [11, Thm. 6.1]. In other words, if the main schema is constrained by EGDs, and the view is a projection, then the view schema has a basis for its constraints consisting entirely of EGDs.

Definition 6.3 (Single relation EGD-schemata and their views and decompositions). While the theory developed here applies to multi-relation schemata, such schemata are constructed by assembling a set of single-relation schemata, each with its own set of EGDs, and then imposing a common set of UINDs. Therefore, it is appropriate to begin with a careful definition of single-relation schemata.

(a) A *single-relation EGD-schema* is a pair $(R[\mathbf{U}], \Phi)$ in which R is a relation name on attribute set $\mathbf{U}$ and Φ is a finite set of EGDs on $R[\mathbf{U}]$.

(b) Given a single relation EGD-schema $\mathbf{R} = (R[\mathbf{U}], \Phi)$ and $\mathbf{W} \subseteq \mathbf{U}$, the *(projection) view defined by* $\mathbf{W}$ is $\Pi_{\mathbf{W}} = ((R[\mathbf{W}], \pi_{\mathbf{W}}(\Phi)), \pi_{\mathbf{W}})$. $\pi_{\mathbf{W}}(\Phi)$ is the projection of the constraints Φ onto $\mathbf{W}$, and $\pi_{\mathbf{W}} : R[\mathbf{U}] \to R[\mathbf{W}]$ is the projection mapping of R from the attributes of $\mathbf{U}$ to the attributes of $\mathbf{W}$.

Although it is often the case that the EGDs of the schema will ensure that the decomposition is lossless (that the EGDs define a lossless decomposition was a requirement in the earlier work [21]), it is not necessary to so require. As expressed in general form in Definition 4.6, the dependency which defines the reconstruction may be specified separately. By its nature, this dependency is not embeddable in the views. In the single-relation case, it is always a join dependency; the following definition makes this explicit.

(c) A *single-relation EGD-schema with JD* is a triple $(R[\mathbf{U}], \Phi, \bowtie[\mathbf{W}_1, \mathbf{W}_2])$ in which $(R[\mathbf{U}], \Phi)$ is a single-relation EGD-schema and $\bowtie[\mathbf{W}_1, \mathbf{W}_2]$ is a full join dependency on $\mathbf{U}$; i.e., $\mathbf{W}_1, \mathbf{W}_2 \subseteq \mathbf{U}$ with $\mathbf{W}_1 \cup \mathbf{W}_2 = \mathbf{U}$. The possibility that either of $\mathbf{W}_1$ and $\mathbf{W}_2$ is empty, or that $\mathbf{W}_1 \cap \mathbf{W}_2 = \emptyset$, is not excluded.

(d) The single-relation EGD-schema with JD $(R[\mathbf{U}], \Phi, \bowtie[\mathbf{W}_1, \mathbf{W}_2])$ *defines a meet-complementary pair* if $\{\Pi_{\mathbf{W}_1}, \Pi_{\mathbf{W}_2}\}$ forms a meet-complementary pair of views of $(R[\mathbf{U}], \Phi)$.

Everything about EGD-schemata is decidable and constructible by algorithm, as shown by the next result.

Proposition 6.4 (Algorithmic construction of covers for EGDs). Let $\mathbf{R} = (R[\mathbf{U}], \Phi, \bowtie[\mathbf{W}_1, \mathbf{W}_2])$ be a single-relation EGD schema with JD which defines a meet-complementary pair $\{\Pi_{\mathbf{W}_1}, \Pi_{\mathbf{W}_2}\}$, and let $k \in \mathbb{N}$.

(a) $\mathbf{R}$ has the $\{\Pi_{\mathbf{W}_1}, \Pi_{\mathbf{W}_2}\}$-relative k-submodel property iff Φ has a basis consisting of EGDs of degree at most k.

(b) It is decidable whether or not $\mathbf{R}$ has the $\{\Pi_{\mathbf{W}_1}, \Pi_{\mathbf{W}_2}\}$-relative k-submodel property.

(c) If $\mathbf{R}$ has the $\{\Pi_{\mathbf{W}_1}, \Pi_{\mathbf{W}_2}\}$-relative k-submodel property, then it is possible to construct a set Ψ of EGDs of degree no more than k with the property that $\Psi \cup \{\bowtie [\mathbf{W}_1, \mathbf{W}_2]\}$ and $\Phi \cup \{\bowtie [\mathbf{W}_1, \mathbf{W}_2]\}$ are equivalent sets of dependencies (for finite or infinite relations).

Proof. To establish (a), let Φ' be the set of all EGDs on $\mathbf{R}$ which are of degree at most k, and which are consequences of those in Φ. Note that Φ' is a finite set, up to renaming of variables, since the number of distinct patterns of equality of variables on the left hand side is finite. Now, let $M \in \mathsf{DB}(\mathbf{D})$ satisfy all dependencies in Φ. Then, M is a $\langle \Gamma_1; \Gamma_2 \rangle$-relative k-submodel, by construction, and hence a model. Thus, Φ' itself must be a basis for Φ, as required.

For (b), let Φ'' be the set of all EGDs on $\mathbf{R}$ which are of degree at most k. This is a finite set, for the same reason as Φ' above is – there are only a finite number of possible patterns for the left hand side of such a constraint. Now, for each such constraint $\varphi \in \Phi''$, simply test to see whether $\Phi \models \varphi$, using any of the standard inference procedures for data dependencies, such as the chase of [4] or the resolution-like procedure of [14]. Since these are all universal dependencies, inference for infinite and finite relations are equivalent. Part (c) follows immediately from this. □

Definition 6.5 (Multi-relation EGD-schemata and their views and decompositions) In part (a) below, a multi-relation EGD schema is nothing more than a suitably presented collection of single-relation EGD schemata. In part (b), a common set of (inter-relational) UINDs is imposed on top of this to obtain the formalization of the schemata to be investigated in this section.

(a) A *multi-relation EGD-schema with JDs* is a set $\{(R_i[\mathbf{U}_i], \Phi_i, \bowtie [\mathbf{W}_{1i}, \mathbf{W}_{2i}]) \mid 1 \leqslant i \leqslant n\}$ in which $(R_i[\mathbf{U}_i], \Phi_i, \bowtie [\mathbf{W}_{1i}, \mathbf{W}_{2i}])$ is a single-relation EGD-schema for $1 \leqslant i \leqslant n$, and the attributes are *pairwise disjoint*, in the precise sense that for $i \neq j$, $\mathbf{U}_i \cap \mathbf{U}_j = \emptyset$.

(b) A *multi-relation EGD-schema with JDs and UINDs* is a pair $(\{(R_i[\mathbf{U}_i], \Phi_i, \bowtie [\mathbf{W}_{1i}, \mathbf{W}_{2i}]) \mid 1 \leqslant i \leqslant n\}, \Phi_{\text{UIND}})$ in which $\{(R_i[\mathbf{U}_i], \Phi_i, \bowtie [\mathbf{W}_{1i}, \mathbf{W}_{2i}]) \mid 1 \leqslant i \leqslant n\}$ is a multi-relation EGD schema with JDs and Φ_{UIND} is a set of UINDs on these relations.

Discussion 6.6 (Axioms of inference for UINDs in the presence of FIDs). As already remarked in summary 6.1, the interaction of FIDs and UINDs is minimal. The following inference rules, from [7]; apply to single-relation schemata constrained by FIDs and UINDs. (Since both EGDs and JDs are FIDs, these rules apply to the context of this paper.) They are presented in the standard format in which the preconditions of the rule are presented above the horizontal line, with the consequences below it.

(a) Reflexivity and transitivity of UINDs:

$$\text{(rt-uid)} \qquad \frac{}{A \sqsubseteq A} \qquad \frac{A \sqsubseteq B \quad B \sqsubseteq C}{A \sqsubseteq C}$$

(b) The *cycle rule* for FDs and UINDs on finite databases:

$$\text{(cy-uid)} \qquad \frac{C_0 \to C_1 \quad C_1 \sqsupseteq C_2 \quad \ldots \quad C_{m-1} \to C_m \quad C_m \sqsupseteq C_0}{C_1 \to C_0 \quad C_2 \sqsupseteq C_1 \quad \ldots \quad C_m \to C_{m-1} \quad C_0 \sqsupseteq C_m}$$

The cycle rule is peculiar to the context of finite databases, and does not apply in the infinite case. It is best understood in terms of a special case. Suppose that both $C_0 \to C_1$ and $C_1 \sqsubseteq C_0$ hold. For a relation r which satisfies these constraints, The FD forces $\mathsf{Card}(\pi_{C_1})(r) \leqslant \mathsf{Card}\pi_{C_0}(r))$, while the UIND forces $\mathsf{Card}(\pi_{C_0})(r) \leqslant \mathsf{Card}(\pi_{C_1}(r)))$. Thus, $\mathsf{Card}(\pi_{C_0})(r) = \mathsf{Card}(\pi_{C_1}(r)))$, and combined with the inclusion implied by the UIND, this forces the two sets to be equal, thus implying $C_1 \to C_0$ and $C_0 \sqsubseteq C_1$. The rule (cy-uid) states this idea in a more general form in which the FDs and UINDs may chain over several relations. This is the *only* way in which FIDs may combine with UINDs to form new constraints of either class. A formalization of these ideas is given next.

Lemma 6.7 (Decoupling of FIDs and UINDs). Let $(R[\mathbf{U}], \Phi)$ be a single-relation schema in which $\Phi = \Phi_{\mathsf{FID}} \cup \Phi_{\mathsf{UIND}}$, with Φ_{FID} a set of FIDs, and Φ_{UIND} a set of UINDs, both taken over the attribute set $\mathbf{U}$. Let φ be any FID or UIND over $R[\mathbf{U}]$.

(a) $\Phi \models_f \varphi$ holds iff $\Phi \vdash \varphi$ may be established using standard inference rules for FIDs (such as the chase [4] or the resolution-like inference procedure of [14]), together with the rules (rt-uid) and (cy-uid) given in Discussion 6.6.

(b) Using this proof procedure, it is decidable whether or not $\Phi \models_f \varphi$ holds.

Here, $\models_f$ denotes finite implication; that is, implication when the models are finite relations, and $\vdash$ denotes deduction via the above-mentioned proof procedures.

Proof. See [7, Corollary 5.3]. Note in particular that this result applies to *finite models*, which is exactly the framework used here. □

Definition 6.8 (Product-relation EGD schemata and EJU/EGD-schemata). The results identified in Lemma 6.7 above apply in the context of single-relation schemata, while the focus of investigation here is multi-relation schemata. The easiest remedy to this problem is to recast (temporarily) the framework of this section into a single-relation format. In this process, a minor additional constraint will be imposed.

The idea is simple – just form the Cartesian product of the component relations using the so-called *cross dependency* $\bowtie [\mathbf{U}_1, \ldots, \mathbf{U}_n]$, which is just a join dependency in which there are no common columns of the component attributes. More precisely, let $\mathbf{R} = (\{(R_i[\mathbf{U}_i], \Phi_i, \bowtie[\mathbf{W}_{1i}, \mathbf{W}_{2i}]) \mid 1 \leqslant i \leqslant n\}, \Phi_{\text{UIND}})$ be a multi-relational EGD-schema with JDs and UINDs.

(a) The *product-relation EGD schema with JDs and UINDs* corresponding to $\mathbf{R}$ is

$$\text{(megd)} \qquad \begin{aligned}&(R_1[\mathbf{U}_1] \times \ldots \times R_n[\mathbf{U}_n],\\ &\left(\bigcup_{i=1}^{n} \Phi_i\right) \cup \Phi_{\text{UIND}} \cup\\ &\qquad \{\bowtie[\mathbf{W}_{1i}, \mathbf{W}_{2i}] \mid 1 \leqslant i \leqslant n\} \cup \{\bowtie[\mathbf{U}_1, \ldots, \mathbf{U}_n]\})\end{aligned}$$

It is important to note in the above that the JDs in $\{\bowtie[\mathbf{W}_{1i}, \mathbf{W}_{2i}] \mid 1 \leqslant i \leqslant n\}$ may be viewed as full (as opposed to embedded) dependencies. This is true because the embedding is into a Cartesian product, and each JD lies in one component of this product. Indeed, define $\overline{\mathbf{W}}_i = \bigcup_{i \neq j}(\mathbf{W}_{1j} \cup \mathbf{W}_{2j})$, and then replace $\bowtie[\mathbf{W}_{1i}, \mathbf{W}_{2i}]$ with $\bowtie[\mathbf{W}_{1i} \cup \overline{\mathbf{W}}_i, \mathbf{W}_{2i} \cup \overline{\mathbf{W}}_i]$. In the presence of $\bowtie[\mathbf{U}_1, \ldots, \mathbf{U}_n]$, these two constraints are equivalent, with the latter being an FID. Thus, all of the constraints, save for those in Φ_{UIND}, are FIDs.

There is a small complication; namely, this approach will not work if some of the relations are empty and others are not. One way around this, taken in [12], is to add a special tuple to each relation, and require that these special tuples always be present. Here, a simpler approach is taken, in which the requirement is imposed that if one relation is nonempty, then they all must be nonempty. In practice, this is hardly a serious drawback, since relations in real databases are almost never empty.

(b) Define the *uniform nonemptiness-emptiness constraint* φ_{UNE} for $\mathbf{R}$ to be that which states that for all $i \in \{1, 2, \ldots, n\}$, it is either the case the the instance of every R_i is empty, or else the case the instance of every R_i is nonempty.

(c) The *uniform nonemptiness-emptiness schema* corresponding to $\mathbf{R}$ is

$$\text{(eju/egd)} \quad (\{(R_i[\mathbf{U}_i], \Phi_i, \bowtie[\mathbf{W}_{1i}, \mathbf{W}_{2i}]) \mid 1 \leqslant i \leqslant n\}, \Phi_{\text{UIND}} \cup \{\varphi_{\text{UNE}}\})$$

(d) A *multi-relation EGD-schema with JDs and UINDs and with the uniform non-emptiness-nonemptiness constraint* is exactly a schema of the above form. For convenience, such a schema will also be called an *EJU/EGD-schema*.

Theorem 6.9 (Constraint interaction on EJU/EGD-schemata). Let $\mathbf{R}$ be an EJU/EGD-schema of the form given in (eju/egd) of Definition 6.8. Then it is possible to construct, by algorithm, finite sets of constraints $\{\Phi'_1, \Phi'_2, \ldots, \Phi'_n, \Phi'_{UIND}\}$ with the following properties.

(a) With $\mathbf{R}'$ defined as

$$(\{(R_i[\mathbf{U}_i], \Phi'_i, \bowtie [\mathbf{W}_{1i}, \mathbf{W}_{2i}]) \mid 1 \leqslant i \leqslant n\}, \Phi'_{UIND} \cup \{\varphi_{UNE}\})$$

it is the case that $\mathsf{LDB}(\mathbf{R}) = \mathsf{LDB}(\mathbf{R}')$.

(b) For each $i \in \{1, 2, \ldots, n\}$, if φ is an EGD on $\mathbf{R}_i$ with $((\bigcup_{i=1}^{n} \Phi) \cup \Phi_{UIND}) \models_f \varphi$, then $\Phi'_i \models_f \varphi$.

(b) If φ is a UIND on $\bigcup_{i=1}^{n} \mathbf{U}_i$ and $((\bigcup_{i=1}^{n} \Phi) \cup \Phi_{\mathrm{UIND}}) \models_f \varphi$, then $\varphi \in \Phi'_{UIND}$.

Proof. Follows directly from Lemma 6.7 and the equivalence between product – and multiple-relational schemata developed in Definition 6.8. The only point of concern is the effect of the join dependencies. However, according to [5, Lemma 7], the join dependencies in $\{\bowtie [\mathbf{W}_{1i} \cup \overline{\mathbf{W}}_i, \mathbf{W}_{2i} \cup \overline{\mathbf{W}}_i] \mid 1 \leqslant i \leqslant n2\} \cup \{\bowtie [\mathbf{U}_1, \ldots, \mathbf{U}_n]\}$ cannot cause any additional EGDs to be generated that would not already be generated by the EGDs alone. Since UINDs and EGDs only interact via FDs, it follows that the JDs are inconsequential in terms of constructing the Φ'_i's and Φ'_{UIND}. □

Notation 6.10 (Notational and constraint-format convention). For the rest of this section, unless stated specifically to the contrary, $\mathbf{R}$ will be taken to be an EJU/EGD-schema of the form shown in (eju/egd) of Definition 6.8. For the sake of convenience, it will also be assumed that the Φ_i's and Φ_{UIND} satisfy the conditions identified in Theorem 6.9 above. In particular, Φ_{UIND} will contain all UINDs which govern the schema.

Also, since the context Φ_{UIND} is fixed, for $A_1, A_2 \in \mathbf{U}$, the statement $A_1 \sqsubseteq A_2$ will be used as an abbreviation for the more cumbersome $A_1 \sqsubseteq A_2 \in \Phi_{\mathrm{UIND}}$.

Notation 6.11 (Notation for the views and decomposition of an EJU/EGD schema). In the general context of the previous sections, the decomposition of the main schema $\mathbf{D}$ was denoted $\{\Gamma_1, \Gamma_2\}$, with the meet of this meet-complementary pair denoted Γ_3. Similarly, a standard notation will be used for the relational case considered here. The component views of the EJU/EGD schema $\mathbf{R}$ will be denoted $\{\Gamma_{\pi_1(\mathbf{R})}, \Gamma_{\pi_2(\mathbf{R})}\}$. These two will be taken to be a meet-complementary pair, with meet $\Gamma_{\pi_3(\mathbf{R})}$. More formally, proceed as follows.

(a) For $i \in \{1, 2\}$, define the schema

$$\pi_i(\mathbf{R}) = \Big\{(R_i[\mathbf{W}_{ij}], \pi_{\mathbf{W}_{ij}}\Big(\bigcup_{m=1}^{n}(\Phi_{im}) \cup \Phi_{\mathrm{UIND}})\Big)1 \mid 1 \leqslant j \leqslant n\Big\}$$

(b) Define the schema

$$\pi_3(\mathbf{R}) = \Big\{(R_i[\mathbf{W}_{1i} \cap \mathbf{W}_{2i}], \pi_{\mathbf{W}_{1i} \cap \mathbf{W}_{2i}}\Big(\bigcup_{m=1}^{n}(\Phi_{im}) \cup \Phi_{\mathrm{UIND}})\Big) \mid 1 \leqslant j \leqslant n\Big\}$$

At this point, the sets of the form $\pi_{\mathbf{Z}}(\bigcup_{m=1}^{n}(\Phi_{im}) \cup \Phi_{\mathrm{UIND}})$ are simply some representation of the constraints which project from $\mathbf{R}$. It remains to characterize their exact nature.

(c) For $i \in \{1,2\}$, the mapping $\pi_i^{\mathbf{R}} : \mathbf{R} \to \pi_i(\mathbf{R})$ is just the collection of the projections $\{\pi_{\mathbf{W}_i} : R[\mathbf{U}_j] \to R[\mathbf{W}_{ij}] \mid 1 \leqslant j \leqslant n\}$. For $i = 3$, $\pi_3^{\mathbf{R}} : \mathbf{R} \to \pi_3(\mathbf{R})$ is $\{\pi_{\mathbf{W}_3} : R[\mathbf{U}_j] \to R[\mathbf{W}_{1j} \cap \mathbf{W}_{2j}] \mid 1 \leqslant j \leqslant n\}$.

(d) For $i \in \{1,2,3\}$, the view $\Gamma_{\pi_i(\mathbf{R})} = (\pi_i(\mathbf{R}), \pi_i^{\mathbf{R}})$.

(e) For $i \in \{1,2\}$, define $\mathbf{U}^i = \bigcup_{j=1}^{n} \mathbf{W}_{ij}$, and define $\mathbf{U}^3 = \mathbf{U}^1 \cap \mathbf{U}^2$. Furthermore, for $i \in \{1,2,3\}$, let $\mathbf{U}^{\bar{i}} = \mathbf{U} \setminus \mathbf{U}^i$,

Notation 6.12 (Further notational convention). Unless stated specifically to the contrary, throughout the remainder of this section, $\{\Gamma_{\pi_1(\mathbf{R})}, \Gamma_{\pi_2(\mathbf{R})}\}$ will be assumed to be a meet-complementary pair, with meet $\Gamma_{\pi_3(\mathbf{R})}$.

Definition 6.13 (The graphs of Φ_{UIND}). The family Φ_{UIND} defines a directed graph in a natural way, with the vertices the attributes in $\mathbf{U}$. An edge from A_1 to A_2 in this graph corresponds to $A_1 \sqsubseteq A_2$, provided that this dependency cannot be inferred by transitive closure. In other words, the graph defines a minimal 'skeleton' of the UINDs, from which the others may be derived via the transitivity rule of (rt-uid) of Discussion 6.6.

(a) Formally, the *graph of* Φ_{UIND}, denoted $\mathsf{Graph}(\Phi_{\mathrm{UIND}})$, is defined as follows.

 (i) The set of vertices of $\mathsf{Graph}(\Phi_{\mathrm{UIND}})$ is just $\mathbf{U}$, the set of all attributes over the relations.

 (ii) For $A_1, A_2 \in \mathbf{U}$, there is a (directed) edge from A_1 to A_2 precisely in the case that $A_1 \sqsubseteq A_2$ and, for all $A_3 \in \mathbf{U}$ with the property that $A_1 \sqsubseteq A_3 \sqsubseteq A_2$, either $A_1 = A_2$ or else $A_2 = A_3$. Write $A_1 \prec A_2$ to indicate that there is an edge from A_1 to A_2 in $\mathsf{Graph}(\Phi_{\mathrm{UIND}})$.

It is important to be able to classify attributes (*qua* vertices) according to the projections in which they lie. The following provides a convenient notation for this.

(b) For $i \in \{1,2,3,\bar{1},\bar{2},\bar{3}\}$ and $A \in \mathbf{U}$, A is an (i)-vertex precisely in the case that $A \in \mathbf{U}^i$.

The graph $\mathsf{Graph}(\Phi_{\mathrm{UIND}})$ may contain cycles, and it is important to have a means of grouping the corresponding vertices together, since they will all have exactly the same projection in any legal database. The reduced graph of Φ_{UIND} collapses $\mathsf{Graph}(\Phi_{\mathrm{UIND}})$ by collecting such equivalent vertices into one.

(c) The vertices $A_1, A_2 \in \mathbf{U}$ are *equivalent* if both $A_1 \sqsubseteq A_2$ and $A_2 \sqsubseteq A_1$ hold. The equivalence class of A_1 under this relation is denoted $[A_1]$.

(d) The *reduced graph of* Φ_{UIND}, denoted $[\mathsf{Graph}](\Phi_{\mathrm{UIND}})$, is defined as follows.

(i) The set of vertices of $\mathsf{Graph}(\Phi_{\mathrm{UIND}})$ is just $\{[A] \mid A \in \mathbf{U}\}$. Denote this set by $[\mathbf{U}]$.

(ii) For $[A_1], [A_2] \in [\mathbf{U}]$, there is a (directed) edge from $[A_1]$ to $[A_2]$ precisely in the case that $[A_1] \neq [A_2]$ and $B_1 \prec B_2$ in $\mathsf{Graph}(\Phi_{\mathrm{UIND}})$ for some $B_1 \in [A_1]$ and $B_2 \in [A_2]$. Write $[A_1] \prec [A_2]$ to indicate that there is an edge from $[A_1]$ to $[A_2]$ in $[\mathsf{Graph}](\Phi_{\mathrm{UIND}})$.

The identification process of (b) above extends to the reduced graph, but two possibilities must be considered. The first occurs when some element from the equivalence class lies in a given region, and the other when all elements from that equivalence class do.

(e) For $i \in \{1, 2, 3, \bar{1}, \bar{2}, \bar{3}\}$ and $[A] \in [\mathbf{U}]$, A is an $(\exists i)$-vertex in the case that some $B \in [A]$ is an (i)-vertex in $\mathsf{Graph}(\Phi_{\mathrm{UIND}})$, and a $(\forall i)$-vertex in the case that every $B \in [A]$ is an (i)-vertex in $\mathsf{Graph}(\Phi_{\mathrm{UIND}})$.

Definition 6.14 (Fanout-free families of UINDs). To realize the main result of this section, it is necessary to place a restriction on the form of allowed family of UINDs. The requirement stipulates that there cannot be branching in the forward direction. Formally, the condition is formulated as follows.

(a) The set Φ_{UIND} is *fanout free* if for any $A_1, A_2, A_3 \in \mathbf{U}$, if both $A_1 \sqsubseteq A_2$ and $A_1 \sqsubseteq A_3$ hold, then one of $A_2 \sqsubseteq A_3$ or $A_3 \sqsubseteq A_2$ holds as well.

It does not appear that restricting consideration to fanout free systems of UINDs is a significant limitation in practice. Consider in particular the ubiquitous *foreign-key constraints* or *referential integrity constraints* [10, Section 5.2.4]. The inclusion $A_2 \sqsubseteq A_1$ may be used to model the situation in which A_2 of (R_2, say) is a foreign key to the primary key A_1 (of R_1, say). There may certainly be another relation (R_3, say) with a foreign key A_3 which also references the primary key A_1 of R_1. Fanout freeness does not prohibit this. What it would prohibit would be a situation in which a foreign key had to be a subset of two distinct primary keys, but this is not part of traditional database modelling.

Of course, primary keys consisting of more than one attribute are allowed in existing database systems, although they are relatively uncommon. These are not modellable directly within the framework developed here, although in real instances of such, the multiple attributes could likely be lumped into one, for the purposes of modelling the dependencies of the schema.

Definition 6.15 (Meet-free traversals and meet-situated ancestors and descendants). The question of which UINDs may be allowed and which must be prohibited in a meet-complementary decomposition is now considered. The only type of UIND which is does not obviously embed into one of the two component views of the decomposition is one which connects an attribute in $\mathbf{U}^{\bar{2}}$ to one in $\mathbf{U}^{\bar{1}}$; that is, one whose two attributes lie in opposite views, with neither lying in the meet. (More precisely, these two attributes cannot be equivalent to any which lie in the meet.) The formalization of this situation is as follows.

(a) The edge $A_1 \prec A_2$ of $\mathsf{Graph}(\Phi_{\mathrm{UIND}})$ defines a *meet-free traversal* from $\Gamma_{\pi_1(\mathbf{R})}$ to $\Gamma_{\pi_2(\mathbf{R})}$ if one of the following two conditions is satisfied.

- (i) $[A_1]$ is a type $(\forall\bar{2})$-vertex and $[A_2]$ is a type $(\forall\bar{1})$-vertex of $[\mathsf{Graph}](\Phi_{\mathrm{UIND}})$.
- (ii) $[A_1] = [A_2]$, A_1 is a $(\bar{2})$-vertex, A_2 is a $(\bar{1})$ vertex, and for all $B \in [A_1]$, B is either a $(\bar{1})$-vertex or else a $(\bar{2})$-vertex of $[\mathsf{Graph}](\Phi_{\mathrm{UIND}})$.

(b) A meet-free traversal from $\Gamma_{\pi_2(\mathbf{R})}$ to $\Gamma_{\pi_1(\mathbf{R})}$ is defined analogously.

The goal is to show that meet-free traversals cannot occur in the case that $\{\Gamma_{\pi_1(\mathbf{R})}, \Gamma_{\pi_2(\mathbf{R})}\}$ is a meet-complementary pair. The idea is simple. If $A_1 \prec A_2$ defines a meet-free traversal from $\Gamma_{\pi_1(\mathbf{R})}$ to $\Gamma_{\pi_2(\mathbf{R})}$, and $M \in \mathsf{LDB}(\mathbf{R})$, then there is an $a \in \pi_{A_1}(M)$ which must necessarily be in $\pi_{A_2}(M)$ as well. Now, replace this a with some $b \in \mathsf{Dom}(\mathbf{U})$ which does not occur in M to obtain a new model M'. Then, consider $M'' = (\pi_1^{\mathbf{R}} \otimes \pi_2^{\mathbf{R}})^{-1}(\pi_1^{\mathbf{R}}(M), \pi_2^{\mathbf{R}}(M'))$ which must also be in $\mathsf{LDB}(\mathbf{R})$, since it is constructed from the appropriate view projections of M and M'. It is clear that M'' cannot satisfy $A_1 \sqsubseteq A_2$, whence a contradiction is obtained. Thus, such a meet-free traversal cannot occur, and so there must be a basis of the UINDs which embed in the views of the decomposition.

Unfortunately, there is a flaw in the above argument. Namely, when a is replaced by b, this may change the state of the meet view $\Gamma_{\pi_3(\mathbf{R})}$, since a may have been visible there as well, due to the presence of other UINDs. Such a change would render the pair $(\pi_1^{\mathbf{R}}(M), \pi_2^{\mathbf{R}}(M'))$ join incompatible, and prevent the construction of M''. Fortunately, there is a way to ensure that this does not happen. The details are now presented.

Informally, a minimal proper meet-situated ancestor of A_1 is a predecessor of A_2 (*i.e.*, $A_2 \sqsubseteq A_1$ which lies in the meet $\Gamma_{\pi_3(\mathbf{R})}$, with the additional constraint that no attribute 'between' between A_2 and A_1 has this property. A minimal proper meet-situated successor is defined dually; i.e., in the opposite direction of inclusion. It should be noted that a minimal proper meet-situated successor must be unique, if it exists, due to the fanout-freeness condition.

In the definitions below, let $i \in \{1, 2\}$, and let $A_1 \in \mathbf{U}^{\bar{i}}$.

(c) A_2 is a *minimal proper meet-situated ancestor* of A_1 if the following conditions are met.

- (i) $A_2 \sqsubseteq A_1$.
- (ii) A_2 is a (3)-vertex; (*i.e.*, A_2 lies in the meet $\Gamma_{\pi_3(\mathbf{R})}$).

(iii) Whenever $A_2 \sqsubseteq A_3 \sqsubseteq A_1$ holds with $[A_1] \neq [A_3] \neq [A_2]$, it must be the case that $[A_3]$ is a $(\forall\bar{\exists})$-vertex; (*i.e.*, A_2 does not lie in the meet $\Gamma_{\pi_3(\mathbf{R})}$).

(d) A_2 is a *minimal proper meet-situated successor* of A_1 if conditions identical to (i)-(iii) above, save that all instances of ‘ $\sqsubseteq$ ’ are replaced with ‘$\sqsupseteq$’, are met.

Now, the first of three lemmata which lead to the construction of the contradictory database M'' of Definition 6.15 is presented.

Lemma 6.16. Assume that Φ_{UIND} is fanout free, let $i \in \{1,2\}$, and let $A_1, A_2 \in \mathbf{U}$ with $A_1 \prec A_2$ a meet-free traversal from $\pi_i(\mathbf{R})$ to $\pi_{2-i}(\mathbf{R})$. Let $M \in \mathsf{DB}(\mathbf{D})$, and let $a \in \mathsf{Dom}(\mathbf{U})$ be such that the following conditions are satisfied on M.

(i) $a \in \pi_{A_1}(M)$.

(ii) For no minimal proper meet-situated ancestor B of A_2 is it the case that $a \in \pi_B(M)$.

(iii) For any minimal proper meet-situated successor B of A_2, there exists $b \in \pi_B(M)$ with $b \notin \pi_{A_2}(M)$.

Then it cannot be the case that $M \in \mathsf{LDB}(\mathbf{R})$.

Proof. To begin, assume that $M \in \mathsf{LDB}(\mathbf{R})$; it will be shown that this assumption is contradictory. The idea is to replace a by b in certain place in M to obtain a new $M' \in \mathsf{LDB}(\mathbf{R})$ with the properties that $\pi_{\mathbf{U}_3}(M) = \pi_{\mathbf{U}3}(M')$ and $M' \in \mathsf{LDB}(\mathbf{R})$ iff $M \in \mathsf{LDB}(\mathbf{R})$. In particular, it will be the case that $\pi_{\mathbf{U}_3}(M) = \pi_{\mathbf{U}_3}(M')$.

First of all, since it is assumed that $M \in \mathsf{LDB}(\mathbf{R})$, it must be the case that $a \in \pi_{A_2}(M)$, since the UIND $A_1 \sqsubseteq A_2$ holds. Now assume that A_2 has a minimal proper meet-situated successor B. In view of the fanout-freeness assumption, all such ancestors must lie in the same equivalence class $[B]$. Choose $b \in \pi_B(M) \setminus \pi_{A_2}(M)$; such an element is guaranteed by (iii) above. Let $[A_2] = [B_1] \prec [B_2] \prec \ldots \prec [B_n] = B$ be the path in $[\mathsf{Graph}](\Phi_{\mathrm{UIND}})$ from $[A_2]$ to $[B]$, and let $[B_i]$ be the last element in this path with $b \notin \mathsf{Dom}(B_i)$. Now, to obtain M' from M, for all attributes $C \in \mathbf{U}$ with $C \sqsubseteq B_i$, replace all occurrences of a with b in the C position of all tuples containing that attribute with value a. (Note that this replacement procedure includes B_i in particular.)

If A_2 does not have a minimal proper meet-situated successor, then simply choose b to be any any domain element which does not already occur in M, and replace all occurrences of a by b in M to obtain M'.

Note four things. First of all, for tuple t and $A \in \mathbf{U}$ in which $t[A]$ is changed from a to b, the value a did not occur as the A-value of any other tuple t' of M. Thus, this replacement cannot force any of the EGDs or the JDs to become unsatisfied, since these constraints are all typed. Within attribute A, this replacement amounts to a simple renaming. Second, all of the UINDs remain satisfied, just by construction.

Third, by the assumption of (ii) above, none of these changes is visible in the meet $\pi_{\mathbf{U}_3}(\mathbf{R})$. Finally, note that this construction works even in the case that $[A_1] = [A_2]$. In this case, the minimal proper meet-situated predecessors and successors will be common to the two attributes, but this will not change the construction.

Thus, $M' \in \mathsf{LDB}(\mathbf{R})$. Since $\pi_3^{\mathbf{R}}(M) = \pi_3^{\mathbf{R}}(M')$, $M'' = (\pi_1^{\mathbf{R}} \otimes \pi_2^{\mathbf{R}})^{-1}(\pi_1^{\mathbf{R}}(M), \pi_2^{\mathbf{R}}(M'))$ is well defined, and must be in $\mathsf{LDB}(\mathbf{R})$ as well. However, it is clear that this latter element violates $A_1 \sqsubseteq A_2$, since $a \in \pi_{A_1}(M'') \setminus \pi_{A_2}(M'')$. This is a contradiction, and so $M \notin \mathsf{LDB}(\mathbf{R})$, as was to be shown. □

Definition 6.17 (Products of databases). Let $\{M_i \mid 1 \leqslant j \leqslant m\} \subseteq \mathsf{DB}(\mathbf{D})$. The *product* of $\prod_{i=1}^{m} M_i$ is obtained by taking the products of the relations com-ponentwise. More specifically, for the relation symbol R_i of $\mathbf{R}$, if $t_i = (a_{i1}, a_{i2}, \ldots, a_{in_i})$ is a tuple in the relation for R_i in M_i, $1 \leqslant i \leqslant m$, then $t = (\langle a_{11}, a_{21}, \ldots, a_{n_i1}\rangle, \langle a_{12}, a_{22}, \ldots, a_{n_i2}\rangle, \ldots, \langle a_{1m}, a_{2m}, \ldots, a_{n_im}\rangle)$ is a tuple in the relation corresponding to R_i in $\prod_{i=1}^{m} M_i$, and conversely. Note that this requires a renaming of the elements of the domain $\mathsf{Dom}(\mathbf{U})$, but this is not a significant issue, since it is the form of the models, and not the names of the domain elements, which are of importance.

In [11], the issues surrounding preservation of models on such products and the corresponding projection operations are studied extensively, and a strong correspondence between such preservation and Armstrong models is established. For the purposes of this paper, only the two simple results stated in the lemma below are needed.

Lemma 6.18.

(a) If $\{M_i \mid 1 \leqslant i \leqslant n\}$ is a finite subset of $\mathsf{LDB}(\mathbf{D})$, then the product $\prod_{i=1}^{n} M_i \in \mathsf{LDB}(\mathbf{D})$ as well.

(b) Let $A_1, A_2 \in \mathbf{U}$ with the property that $A_1 \sqsubseteq A_2 \notin \Phi_{\mathrm{UIND}}$. Then there is an $M \in \mathsf{LDB}(\mathbf{D})$ on which $A_1 \sqsubseteq A_2$ does not hold.

Proof. First of all, part (a) follows almost immediately from [11, Thm. 7.2], in which it is shown that every schema constrained by constraints which are called *extended embedded implicational dependencies (XEIDs)* is closed under products. All of the constraints used here – EGDs, JDs, and UINDs – are special cases of XEIDs. The only caveat is that his result, as stated, holds only on databases for which none of the component relations is empty. However, it is easy to see that it also holds if the database in which all relations are empty is allowed. It is rather the case in which some relations are empty and others are not which causes problems, and that case has already been eliminated from the framework used here via the uniform nonemptiness–emptiness constraint φ_{UNE}.

For part (b), it suffices to note that since $((\bigcup_{i=1}^{n}(\Phi_i \cup \bowtie[\mathbf{W}_{1i}, \mathbf{W}_{2i}])) \cup \Phi_{\mathrm{UIND}}) \not\models_f A_1 \sqsubseteq A_2$, there must be some $M \in \mathsf{DB}(\mathbf{D})$ which satisfies all of the constraints in

$\bigcup_{i=1}^{n}(\Phi_i \cup \bowtie [\mathbf{W}_{1i}, \mathbf{W}_{2i}])$ but does not satisfy $A_1 \sqsubseteq A_2$. This is exactly the M which is needed for part (b). □

Lemma 6.19. For any $A \in \mathbf{U}$ there is an $M \in \mathsf{LDB}(\mathbf{R})$ and an $a \in \pi_A(M)$ with the following properties.

(i) For no $B \in \mathbf{U}$ with $[B] \prec [A]$ is it the case that $a \in \pi_B(M)$.

(ii) For no $B \in \mathbf{U}$ with $[A] \prec [B]$ is it the case that $\pi_B(M) = \pi_A(M)$.

Proof. Let $B \in \mathbf{U}$ with $[B] \prec [A]$; thus, in particular, $B \sqsubseteq A \in \Phi_{\mathrm{UIND}}$ while $A \sqsubseteq B \notin \Phi_{\mathrm{UIND}}$. In view of Lemma 6.18(b), there is an $M_B \in \mathsf{LDB}(\mathbf{R})$ with the property that $A \sqsubseteq B$ is not satisfied by M_B. Thus, there is an $a_B \in \mathbf{U}$ with $a_B \in \pi_A(M_B) \setminus \pi_B(M_B)$. Similarly, for $B \in \mathbf{U}$ with $[A] \prec [B]$; $A \sqsubseteq B \in \Phi_{\mathrm{UIND}}$ while $B \sqsubseteq A \notin \Phi_{\mathrm{UIND}}$, and so there is an $M_B \in \mathsf{LDB}(\mathbf{R})$ with the property that $B \sqsubseteq A$ is not satisfied by M_B. Now, let $S' = \{[B] \mid [B] \prec [A]\} \cup \{[B] \mid [A] \prec [B]\}$ and let S be any subset of S' which selects at least one representative from each equivalence class. Define $M = \prod_{B \in S} M_B$; by Lemma 6.18(a), $M \in \mathsf{LDB}(\mathbf{D})$. Let $a \in M$ be the element whose B^{th} component is a_b. Then $a \notin \pi_B(M)$ for any B with $[B] \prec [A]$, since $a_B \notin M_B$. Hence, M satisfies condition (i) above. Condition (ii) is in fact satisfied by every M_B, and so by the product in particular. □

It is now possible to establish that meet-free traversals cannot exist when the UINDs are fanout free. This in turn leads to the conclusion that a cover of the UINDs embeds in the component views.

Proposition 6.20. If Φ_{UIND} is fanout free, then $\mathsf{Graph}\ (\Phi_{\mathrm{UIND}})$ cannot contain any meet-free traversals.

Proof. Let $i \in \{1,2\}$, and let $A_1, A_2 \in \mathbf{U}$ with $A_1 \prec A_2$ a meet-free traversal from $\pi_i^{\mathbf{R}}$ to $\pi_{2-i}^{\mathbf{R}}$. Let $A = A_1$ in Lemma 6.19, and obtain a database $M \in \mathsf{LDB}(\mathbf{R})$ satisfying the conditions (i) and (ii) of that lemma. Then M also satisfies the three conditions (i)–(ii) of Lemma 6.16, and so cannot be in $\mathsf{LDB}(\mathbf{R})$. This is a contradiction; hence, such a meet-free traversal cannot exist. □

Definition 6.21 (Embedded UINDs). For $i \in \{1,2\}$, let $\pi_i^{\mathbf{R}}(\Phi_{\mathrm{UIND}})$ denote the set of all UINDs which hold on $\pi_i^{\mathbf{R}}$.

Proposition 6.22. If Φ_{UIND} is fanout free, then $\Phi_{\mathrm{UIND}} = (\pi_i^{\mathbf{R}}(\Phi_{\mathrm{UIND}}) \cup \pi_2^{\mathbf{R}}(\Phi_{\mathrm{UIND}}))^+$. In other words, there is an embedded cover of Φ_{UIND} into the two views of the meet complementary pair.

Proof. It is clear that the only UINDs which do not embed into one of the two component views, or which cannot be deduced from such embedded UINDs via application of the transitivity rule of (rt-uid) Discussion 6.6, are those which correspond to meet-free traversals. However, by Proposition 6.20, such traversals cannot occur in the case that Φ_{UIND} is fanout free. □

Finally, the main decomposition result may be established. Just as the inference rules for FIDs and UINDs may be decoupled, so too does the constraint checking on the views of the decomposition decouple into two independent checks, one for the EGDs and the other for the UINDs.

Theorem 6.23 (Decomposition in the presence of EGDs and fanout-free UINDs). Assume that Φ_{UIND} is fanout free, let $k \in \mathbb{N}$, and suppose that for each j, $1 \leqslant j \leqslant n$, Φ_i consists of EGDs of degree at most k. Let $M \in \mathsf{JDB}_{\langle\Gamma_1;\Gamma_2\rangle}(\overline{\mathbf{D}})$. Then $M \in \mathsf{LDB}(\mathbf{R})$ iff for both $i = 1$ and $i = 2$, $\pi_i^{\mathbf{R}}(M)$ satisfies all EGDs in $\pi_i^{\mathbf{R}}(\Phi_i)$ of degree at most k, as well as all UINDs in $\pi_i^{\mathbf{R}}(\Phi_{\mathrm{UIND}})$.

Proof. First of all, consider the same candidate model $M \in \mathsf{JDB}_{\langle\Gamma_1;\Gamma_2\rangle}(\overline{\mathbf{D}})$, but as a database in a modified schema in which the UINDs in Φ_{UIND} are ignored. The resulting schema clearly has the $\{\Gamma_{\pi_1(\mathbf{R})}, \Gamma_{\pi_2(\mathbf{R})}\}$-relative k-submodel property; thus, in view of theorem 5.19, the main theorem on decomposition of schemata with the k-submodel property, to verify that the EGDs in $\bigcup_{i=1}^{n} \Phi_i$ are satisfied, it suffices to check that for both $i = 1$ and $i = 2$, $\pi_i^{\mathbf{R}}(M)$ satisfies all EGDs in $\pi_i^{\mathbf{R}}(\Phi_i)$ of degree at most k.

Now, restore the requirement of checking satisfaction of the UINDs. By Proposition 6.22, a cover of the UINDs must embed into the views. Thus, checking the satisfaction of the members of this cover suffices to verify that the constraints of Φ_{UIND} are satisfied. Hence, $M \in \mathsf{LDB}(\mathbf{D})$. □

Upon restricting the above theorem to the familiar territory of FDs, the following result is obtained. In essence, to check the correctness of a decomposed database, it suffices to verify that each of the component views satisfies all embedded FDs and UINDs.

Corollary 6.24 (Decomposition in the presence of FDs and fanout-free UINDs). Assume that Φ_{UIND} is fanout free, let $k \in \mathbb{N}$, and suppose that for each j, $1 \leqslant j \leqslant n$, Φ_i consists of FDs. Let $M \in \mathsf{JDB}_{\langle\Gamma_1;\Gamma_2\rangle}(\overline{\mathbf{D}})$. Then $M \in \mathsf{LDB}(\mathbf{R})$ iff for both $i = 1$ and $i = 2$, $\pi_i^{\mathbf{R}}(M)$ satisfies all FDs and UINDs from $(\Phi_i)^+ \cup \Phi_{\mathrm{UIND}}$ which embed in the view $\pi_i(\mathbf{R})$.

7. Final remarks

Discussion 7.1 (An alternative approach to recapturing generating constraints). The decomposition results for schemata constrained by comparison constraints (Theorem

5.19) are general and comprehensive. On the other hand, the general results reported for generating constraints (Theorem 5.25) are not as broad. While the results established in Section 6 on schemata constrained by EGDs and fanout-free UINDs nonetheless provide interesting results for what is perhaps the most important type of generating constraint in databases, they make use of very specific properties of the relational data model, and in particular of constraints governed by situations in which finite Armstrong relations exist. A crucial next step in this work is to look for alternative formulations of the general problem which promise to lead to stronger results about decompositions in the presence of generating constraints.

Discussion 7.2 (Strong uniqueness of complements). In the preliminary version of this paper [21], results on the uniqueness of update strategies were also presented. One of the weaknesses of the classical constant-complement strategy, set in the set-based framework, is that complements are almost never unique, and the way in which view updates are reflected back to the main schema depends upon the choice of complement. The principal focus of [20] was to identify situations in which such uniqueness may be obtained. The setting was database schemata with order, and the main result stated that so-called order-based updates, that is, updates which may be realized as a sequence of legal insertions and deletions, have a unique reflection to the main view, independent of the choice of complement [20, 4.3].

In [21, Thm. 5.8], it was shown how to generalize this result in the context of CFA-views. The key idea was to extend update strategies on a view $\Gamma = (\mathbf{V}, \gamma)$ to its syntactic extension $\widehat{\Gamma}$; *i.e.*, from $\mathsf{LDB}(\mathbf{V})$ to $\mathsf{DB}(\mathbf{V})$. Since $\widehat{\Gamma}$ is an order-based view, the results of [20] guarantee that the updates on $\widehat{\Gamma}$ are unique, and so these translate back to Γ. The upshot is that all updates on a CFA-view Γ have a unique translation back to the main schema, provided that they are defined by a constant-complement strategy with some CFA-view Γ' as complement.

Due to space and time restrictions, it was not possible to provide a suitable elaboration to PF-schemata and PF-views in this paper. These uniqueness results will be extended and reported separately.

Discussion 7.3 (Relationship to other work). The problem of characterizing the complexity of the constraints on a view does not appear to have received much attention previously, aside from the classical work of Hull [23], in which it is shown that finite specification is not preserved under projection, and the work of Fagin [11], which includes characterizations of projections of various classes of dependencies.

There has been some work on the complexity of verifying the acceptability of view updates under a constant complement strategy. In an early paper, Cosmadakis and Papadimitriou [8] present pessimistic complexity results for view update under constant complement which would appear to contradict those obtained here. However, they work with general subdirect complements, and not meet complements, and so their results do not apply to the closed update strategies considered here. They also investigate the complexity of identifying a minimal (not necessarily meet) complement which will support a given update, again with pessimistic results.

Recently, Lechtenbörger and Vossen [25] have also looked at the complexity of the problem of identifying (not necessarily meet) complements to views, but for the purpose of identifying information missing in the view, and not with an eye towards update strategies. Their approach, by design, does not concern itself with meet complements or update strategies. Beyond those works, most of the literature on the problem of complexity of view updates is focused on logic databases. The fundamental issues which arise in that context (theory-oriented database models) are quite different from those of instance-oriented database models, and so a meaningful comparison is difficult at best.

Appendix A: A simple counterexample to the finite axiomatizability of relational views

It is part of the folklore of the theory of relational databases that there exists a schema with a simple axiomatization which has a projective view which is not finitely axiomatizable. From time to time, it is useful to be able to point to such an example, complete with an explanation of why finite axiomatizability fails. Unfortunately, they have rarely made it into the literature. Only two are known to the author. In [23, Lemma 4.1], Hull presents an example of a schema with five attributes constrained by three functional dependencies (FDs). The view which is not finitely axiomatizable is a projection onto four of those attributes. In [16], the author identifies a simpler example containing just four attributes and constrained by three FDs, with the corresponding view a projection onto three of those attributes. Unfortunately, that example contains an error; and, in any case, no argument for its validity is offered. In this note, the example of [16] is corrected, and the proof of the lack of finite axiomatizability of the view is elaborated.

Let $\mathbf{E}_1$ be the relational schema with the single relation name $R[ABCD]$ on four attributes; the constraining set of FDs is $\mathcal{F}_1 = \{A \to D, B \to D, CD \to A\}$. The domain of possible values for each attribute is assumed to be infinite. Let $\Pi_{ABC} = (R[ABC], \pi_{ABC})$ denote the view which is the projection onto the attributes ABC. For any $n > 0$, let $r(n)$ denote the instance which is depicted in figure 4. Assume that any two elements with distinct names are distinct values, save that a_1 and a_n may be the same. It is easy to see that $r(n)$ is a legal instance of the main schema if and only if $a_1 = a_n$. Similarly, $r'(n) = \pi_{ABC}(r(n))$ is a legal instance of Π_{ABC} under the implied constraints if and only if $a_1 = a_n$, since a simple 'chase' through any element of $\pi^{-1}_{ABC}(r'(n))$ shows that all of the values in the column of attribute D must be the same. However, if any tuple from $r'(n)$ is deleted, a valid instance of $R[ABC]$ is obtained even if $a_1 \neq a_n$, since it is now possible to have two distinct values appearing in

$$\begin{bmatrix}
a_1 & b_1 & c_1 & d_1 \\
a_1 & b_2 & c_2 & d_1 \\
a_2 & b_2 & c_3 & d_1 \\
a_2 & b_3 & c_4 & d_1 \\
a_3 & b_3 & c_5 & d_1 \\
a_3 & b_4 & c_6 & d_1 \\
a_4 & b_4 & c_7 & d_1 \\
a_4 & b_5 & c_8 & d_1 \\
\vdots & \vdots & \vdots & \vdots \\
a_{k-1} & b_{k-1} & c_{2k-3} & d_1 \\
a_{k-1} & b_k & c_{2k-2} & d_1 \\
\vdots & \vdots & \vdots & \vdots \\
a_{n-1} & b_{n-1} & c_{2n-3} & d_1 \\
a_{n-1} & b_n & c_{2n-2} & d_1 \\
a_n & b_n & c_1 & d_1
\end{bmatrix}$$

Figure 4. The layout of the generic counterexample instance.

column D in an inverse image. This situation is shown in figure 5 with the row containing (a_3, b_4, c_6) deleted. In this case, it need not be the case that $a_1 = a_n$. Therefore, Π_{ABC} is not axiomatizable by any set of sentences having only n free tuple variables (i.e.; a maximum of n variables per column). Since n is arbitrary, Π_{ABC} is not finitely axiomatizable. In particular, it is not axiomatizable by any finite set of equality

$$\begin{bmatrix}
a_1 & b_1 & c_1 & d_1 \\
a_1 & b_2 & c_2 & d_1 \\
a_2 & b_2 & c_3 & d_1 \\
a_2 & b_3 & c_4 & d_1 \\
a_3 & b_3 & c_5 & d_1 \\
\cancel{a_3} & \cancel{b_4} & \cancel{c_6} & \\
a_4 & b_4 & c_7 & d_2 \\
a_4 & b_5 & c_8 & d_2 \\
\vdots & \vdots & \vdots & \vdots \\
a_{k-1} & b_{k-1} & c_{2k-3} & d_2 \\
a_{k-1} & b_k & c_{2k-2} & d_2 \\
\vdots & \vdots & \vdots & \vdots \\
a_{n-1} & b_{n-1} & c_{2n-3} & d_2 \\
a_{n-1} & b_n & c_{2n-2} & d_2 \\
a_n & b_n & c_1 & d_2
\end{bmatrix}$$

Figure 5. The layout of the generic counterexample instance with one row deleted.

generating dependencies (EGDs) [1, 10.1], much less by a family of functional dependencies.

References

[1] S. Abiteboul, R. Hull and V. Vianu, *Foundations of Databases* (Addison-Wesley, 1995).

[2] J. Adámek, H. Herrlich and G. Strecker, *Abstract and Concrete Categories* (Wiley-Interscience, 1990).

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

[4] C. Beeri and M.Y. Vardi, Formal systems for tuple and equality generating dependencies, SIAM Journal on Computing 13(1) (1984) 76–98.

[5] C. Beeri and M.Y. Vardi, A proof procedure for data dependencies, Journal of the Association for Computing Machinery 31(4) (1984) 718–741.

[6] A.K. Chandra and M.Y. Vardi, The implication problem for functional and inclusion dependencies is undedidable, SIAM Journal on Computing 14 (1985) 671–677.

[7] S. Cosmadakis, P.C. Kannelakis and M.Y. Vardi, Polynomial-time implication problems for unary inclusion dependencies, Journal of the Association for Computing Machinery 37(1) (1990) 15–46.

[8] S. Cosmadakis and C. Papadimitriou, Updates of relational views, Journal of the Association for Computing Machinery 31 (1984) 742–760.

[9] B.A. Davey and H.A. Priestly, *Introduction to Lattices and Order*, 2nd edition (Cambridge University Press, 2002).

[10] R. Elmasri and S.B. Navathe, *Fundamentals of Database Systems*, 4th edition (Pearson Education, 2004).

[11] R. Fagin, Horn clauses and database dependencies, Journal of the Association for Computing Machinery 29(4) (1982) 952–985.

[12] R. Fagin and M.Y. Vardi, Armstrong databases for functional and inclusion dependencies, Information Processing Letters 16 (1983) 13–19.

[13] J.N., Foster, M.B. Greenwald, J.T. Moore, B.C. Pierce and A. Schmitt, Combinators for Bi-Directional Tree Transformations: A Linguistic Approach to the View Update Problem, Technical Report MS-CIS-04-15, Department of Computer Science, University of Pennsylvania, 2004. to appear in POPL 2005.

[14] J. Grant and B.E. Jacobs, On the family of generalized dependency constraints, JACM 29(4) (1982) 986–997.

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

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

[17] S.J. Hegner, Characterization of desirable properties of general database decompositions, Annals of Mathematics and Artificial Intelligence 7 (1993) 129–195.

[18] S.J. Hegner, Unique complements and decompositions of database schemata, Journal of Computer and System Sciences 48(1) (1994) 9–57.

[19] S.J. Hegner, 2002, Uniqueness of update strategies for database views, in: *Foundations of Information and Knowledge Systems: Second International Symposium, FoIKS 2002, Salzau Castle, Germany, February 2002, Proceedings*, pp. 230–249.

[20] S.J. Hegner, An order-based theory of updates for database views, Annals of Mathematics and Artificial Intelligence 40 (2004) 63–125.

[21] S.J. Hegner, The relative complexity of updates for a class of database views, in: *Foundations of Information and Knowledge Systems: Third International Symposium, FoIKS 2004, Wilehminenberg Castle, Austria, February 17–20, 2004, Proceedings,* Vol. 2942 of *Lecture Notes in Computer Science*, eds. D. Seipel and J.M. Turull–Torres, (2004) pp. 155–175.

[22] E. Horowitz, S. Sahni and S. Rajasekaran, *Computer Algorithms* (Computer Science, 1998).

[23] R. Hull, Finitely specifiable implicational dependency families, Journal of the Association for Computing Machinery 31(2) (1984) 210–226.

[24] B.E. Jacobs, A.R. Aronson and A.C. Klug, On interpretations of relational languages and solutions to the implied constraint problem, ACM Transactions on Database Systems 7(2) (1982) 291–315.

[25] J. Lechtenbörger and G. Vossen, On the computation of relational view components, ACM Transactions on Database Systems 28 (2003) 175–208.

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

[27] B. Thalheim, *Dependencies in Relational Databases*, Vol. 126 of *Teubner-Texte zur Mathematik* (Teubner, 1991).

[28] K. Wang and M.H. Graham, Constant-time maintainability: A generalization of independence, ACM Transactions on Database Systems 17(2) (1992) 201–246.