Journal of Universal Computer Science, vol. 16, no. 20 (2010), 2956-2985
submitted: 7/1/10, accepted: 13/9/10, appeared: 1/11/10 © *J.UCS*

Internal Representation of Database Views

Stephen J. Hegner
(Umeå University, Department of Computing Science
SE-901 87 Umeå, Sweden
hegner@cs.umu.se   http://www.cs.umu.se/~hegner)

Abstract: Although a database view embodies partial information about the state of the main schema, the state of the view schema is a quotient (and not a subset) of the state of the main schema. It is the *information content* of the view state, the set of sentences which are true for that state, and not the state itself which is a subset of the information content of the state of the main schema. There are thus two dual approaches to modelling this partiality, one based upon structures, with a consequent quotient relationship, and another based upon logical theories, with a consequent subset relationship. In this work, a representation for database views is developed which combines these two approaches. The state-based representation is expanded so that the information content embodied in a wide class of views, including those defined by SPJ queries, is fully representable, thus permitting the view state to be modelled internally as a subset of the main database state. The utility of this framework is demonstrated with a simple solution to the uniqueness problem for view updates via constant complement.
Key Words: information, database, modelling, view
Category: H.1.1, H.2.1

1 Introduction

Views are ubiquitous in information systems. It is rare that a single user or application has access to all of the information which is represented in the system. Rather, partial information is provided through views. There are two, dual ways of representing this partiality. In the state-based approach to representing a view $\Gamma = (\mathbf{V}, \gamma)$ of a relational schema $\mathbf{D}$, the defining entity is a surjective function $\gamma_{\mathsf{LDB}} : \mathsf{LDB}(\mathbf{D}) \to \mathsf{LDB}(\mathbf{V})$ which maps the legal databases $\mathsf{LDB}(\mathbf{D})$ of the main schema to the legal databases $\mathsf{LDB}(\mathbf{V})$ of the view schema $\mathbf{V}$. The partiality is embodied in the fact that γ_{LDB} is not in general injective; rather, many states of $\mathbf{D}$ may map to the same view state. The formal representation of partiality is consequently a quotient construction, defined specifically by the *congruence* $\mathsf{Congr}(\gamma)$ on $\mathsf{LDB}(\mathbf{D})$ with $(M_1, M_2) \in \mathsf{Congr}(\gamma)$ iff $\gamma_{\mathsf{LDB}}(M_1) = \gamma_{\mathsf{LDB}}(M_2)$.

Dual to the state-based approach is the information-based approach, in which the "state" of a schema is represented not by a single model (i.e., a single member of $\mathsf{LDB}(\mathbf{D})$), but rather by a set of sentences which define constraints on what the state may be. Formally, the mapping from view sentences to sentences on the main schema is obtained by regarding the view mapping γ as a query in the relational calculus. The fact that this query must return the tuples of the view defines logical constraints which must be true on the state of the main schema. These constraints define a subset of all possible states of the main schema — those which map to the given view state, and so provide a characterization of that view state. More fundamentally, the information-based approach has its merits as an approach to modelling; indeed, it forms the foundation of deductive databases and logic programming [Minker, 1987]. On the other hand, it also carries a great deal of overhead in that a potentially complex set of constraints on what the state may be replaces the state itself in the representation. Nevertheless, even if there is no need to model extensive incomplete information in the main schema itself, it is often very useful and natural to be able to regard the view schema as a subschema of the main schema, and such an information-based approach seems essential to realize that end. The goal of this paper is to address that desideratum by pursuing a rapprochement of the state-based approach and the constraint-based approach, in which the essence of the state-based approach is retained while admitting enough of the constraint-based approach to allow the view schema to be regarded as a subschema of the main schema, A simple example will help illustrate the main ideas.

Let $\mathbf{E}_1$ be the relational schema with the single ternary relation symbol $R[ABC]$, constrained by the join dependency $\bowtie[AB, BC]$. Let $\Pi_{AB}^{\mathbf{E}_1} = (\mathbf{W}_{11}, \pi_{AB}^{\mathbf{E}_1})$ be the view whose schema $\mathbf{W}_{11}$ has the single relation symbol $R_{AB}[AB]$, with $\pi_{AB}^{\mathbf{E}_1}$ the projection of $R[ABC]$ onto $R_{AB}[AB]$; thus $(\forall x_A)(\forall x_B)(R_{AB}(x_A, x_B) \Leftrightarrow (\exists z)(R(x_A, x_B, z)))$. Define the view $\Pi_{BC}^{\mathbf{E}_1} = (\mathbf{W}_{12}, \pi_{BC}^{\mathbf{E}_1})$ similarly as the projection of $R[ABC]$ onto $R_{BC}[BC]$. Consider the state $N_{11} = \{R_{AB}(\mathrm{a}_1, \mathrm{b}_1), R_{AB}(\mathrm{a}_2, \mathrm{b}_2)\}$ of $\mathbf{W}_{11}$. Rather than represent it as the equivalence class $\{M \in \mathsf{LDB}(\mathbf{D}) \mid (\pi_{AB}^{\mathbf{E}_1})_{\mathsf{LDB}}(M) = N_{11}\}$, as in the quotient approach, it is represented as the Boolean conjunctive query $(\exists x_1)(R(\mathrm{a}_1, \mathrm{b}_1, x_1)) \wedge (\exists x_2)(R(\mathrm{a}_2, \mathrm{b}_2, x_2))$. Thus, a generalized state of the main schema becomes a Boolean conjunctive query. Conventional ground atoms are a special case of such queries. Since all variables are quantified existentially, the quantifiers may be dropped, as long as the convention is adopted that all quantified variables are distinct. Thus, this state may be represented as the *atombase* $M_{12} = \{R(\mathrm{a}_1, \mathrm{b}_1, x_1), R(\mathrm{a}_2, \mathrm{b}_2, x_2)\}$. There is a natural correspondence between atombases — finite sets of (not necessarily ground) atoms — and Boolean conjunctive queries. It is a representation of view states based upon this correspondence which is developed within this work. The *internal representation* to $\mathbf{E}_1$ of the view mapping $\pi_{AB}^{\mathbf{E}_1}$ is $\overrightarrow{\pi_{AB}^{\mathbf{E}_1}} : \mathbf{E}_1 \to \mathbf{E}_1$ defined on tuples by $R(\mathrm{a}, \mathrm{b}, \mathrm{c}) \mapsto R(\mathrm{a}, \mathrm{b}, x))$, with a distinct variable x used for each tuple. In other words, the C-attribute value of each R-tuple is replaced by a distinct variable.

The reader may feel that this approach is nothing more than a clever use of null values, and, certainly, there are similarities. However, a Boolean query is simply a statement about what is true in a database state. To illustrate the differences, observe that the atombases $\{R(\mathrm{a},\mathrm{b},x_1), R(\mathrm{a},\mathrm{b},x_2), R(x_3,x_4,x_5)\}$ and $\{R(\mathrm{a},\mathrm{b},x_6)\}$ are completely equivalent; each embodies precisely the information generated by $(\exists x)(R(\mathrm{a},\mathrm{b},x))$. Several copies of a sentence equivalent to $(\exists x)(R(\mathrm{a},\mathrm{b},x))$ embodies no more information than a single copy. As a more salient example, consider the state $M_{11} = \{R(\mathrm{a},\mathrm{b},x_1), R(x_2,\mathrm{b},\mathrm{c})\}$ of $\mathbf{E}_1$. Since the schema is governed by the join dependency $\bowtie [AB, BC]$, M_{11} is equivalent to $M_{12} = \{R(\mathrm{a},\mathrm{b},\mathrm{c})\}$; i.e., x_1 must equal c and x_2 must equal a. That x_1 (resp. x_2) represents some value distinct from c (resp. a) is not representable in this framework. Indeed, that $\mathbf{E}_1$ decomposes losslessly into $\Pi^{\mathbf{E}_1}_{AB}$ and $\Pi^{\mathbf{E}_1}_{BC}$ with the join as the reconstruction mapping recaptured within this framework by the fact that the state M_{12} of $\mathbf{E}_1$ is equivalent to the combined information of its image $\{R(\mathrm{a},\mathrm{b},x_1)\}$ under $\overrightarrow{\pi^{\mathbf{E}_1}_{AB}}$ and its image $\{R(x_2,\mathrm{b},\mathrm{c})\}$ under $\overrightarrow{\pi^{\mathbf{E}_1}_{BC}}$.

The main result of this paper is a systematic development of these ideas. The context imposes two main requirements. First, the view must be defined by Boolean conjunctive queries (i.e., SPJ-queries in the traditional relational terminology). Second, the main schema must admit a chase procedure which always terminates, so that the information reflected from the view state can be completed to a least atombase (representing a Boolean conjunctive query) of the main schema which satisfies its constraints. Under these conditions, it is shown that every such view has an internal representation within the main schema, in the precise sense that every view state has a canonical representation as an atombase of the main schema.

The paper is organized as follows. In Section 2, the relational and logical background upon which the work is founded is presented. In Section 3, the ideas surrounding atombases — compact representations of databases defined by Boolean conjunctive queries — are developed. Section 4 contains the main results of the paper, the characterization of embeddable views. In Section 5, it is shown that the set of canonical atombases for a given schema admit a natural lattice structure. In Section 6, these ideas are applied to the uniqueness problem for view updates via constant complement. Finally, in Section 7, some conclusions and further directions are provided. An index of terminology and notation may also be found just before the list of references.

2 Background

The framework developed in this paper is based upon the classical relational model, and familiarity with its fundamental ideas, as presented in monographs such as [Paredaens et al., 1989] and [Abiteboul et al., 1995], is assumed. The purpose of this section is to lay out some special notions and terminology, within that framework, which will be useful in this work. Familiarity with the fundamentals of first-order logic, as presented in, for example [Monk, 1976], is also assumed.

The notions presented below have a common base with that of earlier papers of the author, including particularly [Hegner, 2008a], but the requirements of the particular problem studied here have mandated numerous changes.

Definition 1 (Relational contexts). A *relational context* contains the logical information which is shared amongst the schemata and database mappings. Formally, a relational context consists of a finite nonempty set Attr of attribute names, a countable set Vars of variables, and a countable set Const of constant symbols. The variables in Vars are further partitioned into two disjoint sets; a countable set $\mathsf{GVars} = \{x_0, x_1, x_2, \ldots\}$ of *general variables*, and a special Attr-indexed set $\mathsf{AttrVars} = \{x_A \mid A \in \mathsf{Attr}\}$ of *attribute variables.* The latter is used only in the definition of interpretation mappings; see Definition 8 for details. Lowercase letters at the end of the alphabet, such as v, w, x, y, and z, as well as subscripted instances using these names, will also be used as general variables.

The constants are the usual domain values; thus, Const is the set of values which a term for an attribute A may assume. There is no partitioning of the constants on an attribute-by-attribute basis; any constant may be the value for any attribute. This choice is made in support of the theory which is developed while keeping the complexity of the framework within bounds. Distinct constants are never equal to one another. This is made explicit by the unique naming constraint (UNA) [Genesereth and Nilsson, 1987, p. 120], which always holds and which is formalized as $(\neg(\mathrm{a} = \mathrm{b}))$ for distinct $\mathrm{a}, \mathrm{b} \in \mathsf{Const}$.

A relational context $\mathcal{D}$ is fixed throughout this paper.

Definition 2 (Tuples and databases). An *unconstrained relational schema* is a pair $\mathbf{D} = (\mathsf{Rels}(\mathbf{D}), \mathsf{Ar}_{\mathbf{D}})$ in which $\mathsf{Rels}(\mathbf{D})$ is a finite set of relational symbols and $\mathsf{Ar}_{\mathbf{D}} : \mathsf{Rels}(\mathbf{D}) \to \mathbf{2}^{\mathsf{Attr}}$ a function which assigns a set of distinct attributes from Attr, to each $R \in \mathsf{Rels}(\mathbf{D})$.

An *R-tuple* is any tuple indexed by the attributes of R. More formally, it is a function with domain $\mathsf{Ar}_{\mathbf{D}}(R)$. An *$R$-atom* is an R-tuple whose terms lie in $\mathsf{Const} \cup \mathsf{Vars}$; more precisely, it is a function $t : \mathsf{Ar}_{\mathbf{D}}(R) \to \mathsf{Const} \cup \mathsf{Vars}$ with the property that for each $A \in \mathsf{Ar}_{\mathbf{D}}(R)$, $t[A] \in \mathsf{Const} \cup \mathsf{GVars} \cup \{x_A\}$; A *$\mathbf{D}$-atom* (or just an atom if $\mathbf{D}$ is clear from the context) is an R-atom for some $R \in \mathsf{Rels}(\mathbf{D})$. A *ground atom* has the property that it contains no variables. The set of all atoms (resp. ground atoms) is denoted $\mathsf{Atoms}(\mathbf{D})$ (resp. $\mathsf{GndAtoms}(\mathbf{D})$).

Databases are modelled as finite sets of ground atoms. A finite set of complete ground $\mathbf{D}$-atoms is called a *database* for $\mathbf{D}$. The set of all databases for $\mathbf{D}$ is denoted $\mathsf{DB}(\mathbf{D})$. Although not databases in the strict sense, finite sets of atoms whose terms may include variables will also arise in this work, as already suggested by the examples of Section 1. A finite subset of $\mathsf{Atoms}(\mathbf{D})$ is called an *atombase* for $\mathbf{D}$. The set of all atombases is denoted $\mathsf{AB}(\mathbf{D})$.

To keep a handle on the notation for various finite sets of atoms, names of the form $\mathsf{xxxDB}(\mathbf{D})$ always identify *databases*; that is, sets of atoms which do not contain any variables, while names of the form $\mathsf{xxxAB}(\mathbf{D})$ always identify *atombases*; that is, sets of atoms which may contain variables.

Definition 3 (Equality atoms). There is a third type of atom which will be of use in defining constraints, the equality atom. Formally, an *equality atom* is of one of the forms $(x_i = x_j)$, $(x_i = \mathrm{a}_j)$, or $(\mathrm{a}_i = \mathrm{a}_j)$, for $x_i, x_j \in \mathsf{GVars}$ and $\mathrm{a}_i, \mathrm{a}_j \in \mathsf{Const}$. The set of all equality atoms is denoted $\mathsf{EqAtoms}$. Equality atoms which equate two constants; e.g., $(\mathrm{a}_i = \mathrm{a}_j)$ are called *ground equality atoms*; note that the truth value of such atoms is predetermined by the unique naming assumption. All other equality atoms; e.g., those of the forms $(x_i = x_j)$ or $(x_i = \mathrm{a}_j)$, are called *variable equality atoms*. The set of all variable equality atoms is denoted $\mathsf{VarEqAtoms}$. The definition of equality atom depends only upon the relational context $\mathcal{D}$, and not upon the specific schema $\mathbf{D}$.

Definition 4 (Formulas and constraint classes). The first-order language associated with the relational schema $\mathbf{D}$ is defined in the natural way; however, it is useful to introduce some notation which identifies particular sets of formulas. Define $\mathsf{WFF}(\mathbf{D})$ to be the set of all well-formed first-order formulas with equality in the language whose set of relational symbols is $\mathsf{Rels}(\mathbf{D})$, whose variables are those of Vars, whose set of constant symbols is Const, and which contains no non-nullary function symbols. The formulas are typed only to the extent that for $A \in \mathsf{Attr}$, a term for A must lie in $\mathsf{Vars} \cup \{x_A\} \cup \mathsf{Const}$.

$\mathsf{WFF}(\mathbf{D}, \exists\wedge+)$ is the subset of $\mathsf{WFF}(\mathbf{D})$ in which only existential quantification is allowed, and the only logical connective which is allowed is conjunction $(\wedge)$; in particular, negation is disallowed, defining the *conjunctive queries* [Chandra and Merlin, 1977]. These classes may be limited to sentences; i.e., formulas without free variables; notationally WFS replaces WFF. For example, $\mathsf{WFS}(\mathbf{D}, \exists\wedge+)$ is the set of sentences in $\mathsf{WFF}(\mathbf{D}, \exists\wedge+)$.

The subscript $\neq$ will be used to denote that equality atoms are not allowed. Thus, for example, $\mathsf{WFS}_{\neq}(\mathbf{D}, \exists\wedge+)$ denotes the subset of $\mathsf{WFS}(\mathbf{D}, \exists\wedge+)$ whose formulas do not contain equality atoms. The set $\mathsf{WFS}_{\neq}(\mathbf{D}, \exists\wedge+)$ occurs so frequently in this paper that it is useful to introduce an abbreviated notation for it; $\varUpsilon^{\mathbf{D}}$ will be used as an abbreviation for $\mathsf{WFS}_{\neq}(\mathbf{D}, \exists\wedge+)$. It will further be assumed that the elements of $\varUpsilon^{\mathbf{D}}$ are in prefix-matrix form with no nesting. Thus, every element of $\varUpsilon^{\mathbf{D}}$ is of the form $(\exists x_1)(\exists x_2)\ldots(\exists x_k)(A_1 \wedge A_2 \wedge \ldots \wedge A_n)$ with each $A_i \in \mathsf{Atoms}(\mathbf{D})$.

Every sentence in $\mathsf{WFS}(\mathbf{D}, \exists\wedge+, \mathsf{Const})$ is equivalent to one in $\varUpsilon^{\mathbf{D}}$. Indeed, if $\varphi \in \mathsf{WFS}(\mathbf{D}, \exists\wedge+, \mathsf{Const})$ contains a conjunct of the form $(x_i = x_j)$, just replace x_j with x_i everywhere and drop the quantifier $(\exists x_j)$. Similarly, if it contains a conjunct of the form $(\mathrm{a} = x)$ replace each occurrence of x with a and drop the quantifier $(\exists x)$. If it contains a conjunct of the form $(\mathrm{a}_1 = \mathrm{a}_2)$, then in view of the unique naming assumption, the formulas must be false unless a_1 and a_2 are the same symbol, in which case that conjunct may simply be dropped. Thus, there is no loss of generality in excluding equality atoms from $\Upsilon^{\mathbf{D}}$.

The symbol $\perp$ denotes the sentence which is always false.

Definition 5 (Complete atomic models). In this work, databases are sets of ground atoms, and not models of sentences in the usual sense. Nevertheless, it is important to have a notion of model. To this end, in a manner analogous to [Monk, 1976, Def. 19.8], for $M \in \mathsf{DB}(\mathbf{D})$, define the *diagram* of M to be $\mathsf{Diagram}(M) = M \cup \{\neg\psi \mid \psi \in \mathsf{GndAtoms}(\mathbf{D}) \setminus M\}$. Say that M is a *complete information model* of $\varphi \in \mathsf{WFS}(\mathbf{D})$ if $\mathsf{Diagram}(M) \cup \{\varphi\} \cup \mathsf{UNA}$ is a consistent set of sentences in $\mathsf{WFS}(\mathbf{D})$.

$\mathsf{DBMod}_{\mathbf{D}}(\varphi)$ denotes the set of all complete information models of φ, with $\mathsf{DBMod}_{\mathbf{D}}(\Phi) = \bigcap\{\mathsf{DBMod}_{\mathbf{D}}(\varphi) \mid \varphi \in \Phi\}$ for $\Phi \subseteq \mathsf{WFS}(\mathbf{D})$. For $\Phi_1, \Phi_2 \subseteq \mathsf{WFS}(\mathbf{D})$, the notation $\Phi_1 \models \Phi_2$ means that $\mathsf{DBMod}_{\mathbf{D}}(\Phi_2) \subseteq \mathsf{DBMod}_{\mathbf{D}}(\Phi_1)$, with $\Phi_1 \models\!\!\!\mid \Phi_2$ denoting that both $\Phi_1 \models \Phi_2$ and $\Phi_2 \models \Phi_1$ hold; i.e., that $\mathsf{DBMod}_{\mathbf{D}}(\Phi_2) = \mathsf{DBMod}_{\mathbf{D}}(\Phi_1)$. For a single $\varphi \in \mathsf{WFS}(\mathbf{D})$, set brackets will often be omitted. Thus, $\mathsf{DBMod}_{\mathbf{D}}(\varphi)$ denotes $\mathsf{DBMod}_{\mathbf{D}}(\{\varphi\})$, and notations such as $\Phi \models \varphi$ and $\Phi \models\!\!\!\mid \varphi$ have the obvious meanings of $\Phi \models \{\varphi\}$ and $\Phi \models\!\!\!\mid \{\varphi\}$.

Notation 6 (Extracting constants and variables). For any $\Phi \subseteq \mathsf{WFS}(\mathbf{D})$, $\mathsf{ConstOf}(\Phi)$ (resp. $\mathsf{VarsOf}(\Phi)$) denotes the set of all constants (resp. variables) which occur in Φ. For $\varphi \in \mathsf{WFS}(\mathbf{D})$, $\mathsf{ConstOf}(\varphi) = \mathsf{ConstOf}(\{\varphi\})$ and $\mathsf{VarsOf}(\varphi) = \mathsf{VarsOf}(\{\varphi\})$. Finally, continuing with the notation above, $\mathsf{TermsOf}(\Phi) = \mathsf{VarsOf}(\Phi) \cup \mathsf{ConstOf}(\Phi)$ with $\mathsf{TermsOf}(\varphi) = \mathsf{TermsOf}(\{\varphi\})$.

Definition 7 (Schemata with constraints and constrained databases). To obtain full relational schemata, constraints are added to the unconstrained schemata of Definition 2. Formally, a *relational schema* is a triple $\mathbf{D} = (\mathsf{Rels}(\mathbf{D}), \mathsf{Ar}_{\mathbf{D}}, \mathsf{Constr}(\mathbf{D}))$ in which $(\mathsf{Rels}(\mathbf{D}), \mathsf{Ar}_{\mathbf{D}})$ is an unconstrained relational schema and $\mathsf{Constr}(\mathbf{D}) \subseteq \mathsf{WFS}(\mathbf{D}, \mathsf{Const})$ is a set of *dependencies* or *constraints* of $\mathbf{D}$. It will further be assumed that $\mathsf{ConstOf}(\mathsf{Constr}(\mathbf{D}))$ is finite; that is, the number of constant symbols which occur in $\mathsf{Constr}(\mathbf{D})$ is finite. This condition will automatically be satisfied whenever $\mathsf{Constr}(\mathbf{D})$ is a finite set. The reason for allowing $\mathsf{Constr}(\mathbf{D})$ to be an infinite set is related to the issue of characterizing the constraints on view schemata; see Definition 8 for a further discussion.

$M \in \mathsf{DB}(\mathbf{D})$ is called a *legal database* for $\mathbf{D}$ if $M \in \mathsf{DBMod}_{\mathbf{D}}(\mathsf{Constr}(\mathbf{D}))$. The set of all legal databases of $\mathbf{D}$ is denoted $\mathsf{LDB}(\mathbf{D})$. Constraint satisfaction for atombases is elaborated in Definition 23.

Definition 8 (External LDB-Views). There are two complementary ways in which a relational view may be presented. Consider again the schema $\mathbf{E}_1$ and the view $\Pi^{\mathbf{E}_1}_{AB}$ of Section 1. The view mapping may be expressed, on the one hand, as a function from $\mathsf{LDB}(\mathbf{E}_1)$ to $\mathsf{LDB}(\mathbf{W}_{11})$, expressed in the relational algebra as the function which maps each tuple in $M \in \mathsf{LDB}(\mathbf{E}_1)$ to its projection on AB. On the other hand, it may also be described in the relational calculus as the query $(\exists z)(R(x_A, x_B, z))$ on $\mathbf{E}_1$. In this work, both representations are of central importance, although it is the representation via the relational calculus which requires a careful presentation. It rests more formally upon the notion of logical interpretation [Jacobs et al., 1982].

A view of $\mathbf{D}$ is a pair $\Gamma = (\mathbf{V}, \gamma)$ in which $\mathbf{V}$ is a database schema and $\gamma : \mathbf{D} \to \mathbf{V}$ is a database morphism which is surjective in a sense to be made precise shortly. Given $R \in \mathsf{Rels}(\mathbf{V})$, an *interpretation* for R into $\mathbf{D}$ is a formula $\varphi \in \mathsf{WFF}(\mathbf{D}, \mathsf{Const})$ in which precisely the variables $\{x_A \mid A \in \mathsf{Ar}_{\mathbf{D}}(R)\}$ are free, with x_A bound to attribute A. The set of all interpretations of R into $\mathbf{D}$ is denoted $\mathsf{Interp}(R, \mathbf{D})$. An *interpretation family* for $\mathbf{V}$ into $\mathbf{D}$ is given by an interpretation formula for each $R \in \mathsf{Rels}(\mathbf{V})$. More formally, think of a view morphism $\gamma : \mathbf{D} \to \mathbf{V}$ as defined by the family $\{\gamma^R \mid R \in \mathsf{Rels}(\mathbf{V})$ and $\gamma^R \in \mathsf{Interp}(R, \mathbf{D})\}$. In the above example, $(\pi^{\mathbf{E}_1}_{AB})^{R_{AB}} = (\exists z)(R(x_A, x_B, z))$.

Let $R \in \mathsf{Rels}(\mathbf{V})$ and let $t \in \mathsf{Atoms}(\mathbf{V})$ be an R-atom. The *tuple substitution* of t into γ, denoted $\mathsf{Substf}\langle \gamma, t\rangle$, is the formula in $\mathsf{WFF}(\mathbf{D})$ obtained by substituting $t[A]$ for x_A in γ^R for each $A \in \mathsf{Ar}_{\mathbf{D}}(R)$. If $t \in \mathsf{GndAtoms}(\mathbf{V})$, then $\mathsf{Substf}\langle \gamma, t\rangle \in \mathsf{WFS}(\mathbf{D})$; i.e., it is a sentence. For $M \in \mathsf{AB}(\mathbf{D})$, define

$$\gamma_{\mathsf{AB}}(M) = \{t \in \mathsf{Atoms}(\mathbf{V}) \mid M \in \mathsf{DBMod}_{\mathbf{D}}(\mathsf{Substf}\langle \gamma, t\rangle)\}$$

If $\gamma_{\mathsf{AB}}(M) \in \mathsf{LDB}(\mathbf{V})$ for each $M \in \mathsf{LDB}(\mathbf{D})$, call γ an *LDB-morphism*. In this case, define $\gamma_{\mathsf{LDB}} : \mathsf{LDB}(\mathbf{D}) \to \mathsf{LDB}(\mathbf{V})$ to be the appropriate restriction of γ_{AB}. Here γ_{LDB} is just the usual mapping which sends states of the main schema to states of the view. Call γ *LDB-surjective* if γ_{LDB} is a surjective function. An *external LDB-view* of $\mathbf{D}$ is a pair $\Gamma = (\mathbf{V}, \gamma)$ in which $\mathbf{V}$ is a database schema and $\gamma : \mathbf{D} \to \mathbf{V}$ is an LDB-morphism which is LDB-surjective. Without further qualification, the term *view* will always mean external view.

If $\gamma^R \in \mathsf{WFF}(\mathbf{D}, \exists\wedge+)$ for each $R \in \mathsf{Rels}(\mathbf{D})$, then γ is called a morphism of *class* $\exists\wedge+$. The external view $\Gamma = (\mathbf{V}, \gamma)$ is defined to be of class $\exists\wedge+$ precisely in the case that γ has that property. The usual SPJ-queries (select, project, join) are all of class $\exists\wedge+$, with the restriction that select queries must be on a single value for each attribute involved in the select. Call Γ *variable normalized* if for distinct $R_1, R_2 \in \mathsf{Rels}(\mathbf{D})$, γ^{R_1} and γ^{R_2} have no quantified variables in common. A view may always be variable normalized by renaming variables.

It is well known that even for simple projective views of relational schemata constrained only by a few functional dependencies, a basis for the implied constraints on the view schema may be infinite [Hull, 1984, Lem. 4.1] [Hegner, 2006, App. A]. Thus, the set $\mathsf{Constr}(\mathbf{V})$ of constraints on the view schema $\mathbf{V}$ is allowed to be infinite in general, although it will be required that $\mathsf{ConstOf}(\mathsf{Constr}(\mathbf{D}))$ be finite. This is of little consequence in this work since the constraints on view schemata are never considered explicitly.

Definition 9 (Closure and minimal covers). Let $\Phi \subseteq \Psi \subseteq \mathsf{WFS}(\mathbf{D})$. The *closure* of Φ in Ψ, denoted $\mathsf{Closure}\langle\Phi,\Psi\rangle$, is $\{\varphi \in \Psi \mid \Phi \models \varphi\}$. A *cover* for Φ relative to Ψ is a subset $\Phi' \subseteq \Psi$ with $\mathsf{Closure}\langle\Phi',\Psi\rangle = \mathsf{Closure}\langle\Phi,\Psi\rangle$. A *minimal cover* Ψ' has the property that none of its proper subsets is itself a cover. Φ is said to be *finitely generated* in Ψ it admits a finite minimal cover. Note that if Ψ is closed under conjunction and Φ is finitely generated, there must be a single $\varphi \in \Psi$ which is a (necessarily minimal) cover for Φ.

3 Information Content and Canonical Models

Notation 10. Throughout this section, unless stated explicitly to the contrary, take $\mathbf{D}$ to be an unconstrained database schema.

Definition 11 (Information content). The *information content* of $M \in \mathsf{DB}(\mathbf{D})$ is defined to be $\mathsf{Info}_{\mathbf{D}}\langle M\rangle = \{\varphi \in \Upsilon^{\mathbf{D}} \mid M \in \mathsf{DBMod}_{\mathbf{D}}(\varphi)\}$. Viewing each element of $\Upsilon^{\mathbf{D}}$ to be a Boolean-valued query, $\mathsf{Info}_{\mathbf{D}}\langle M\rangle$ is exactly the set of such queries which are true on M.

For a single database, this definition is rather trivial, since $\mathsf{Info}_{\mathbf{D}}\langle M\rangle$ is just $\mathsf{Closure}\langle M,\Upsilon^{\mathbf{D}}\rangle$. It is of greater use when the information common to a set of databases is considered. A key example is the following. Let $\Gamma = (\mathbf{V},\gamma)$ be a view of $\mathbf{D}$. For each $N \in \mathsf{LDB}(\mathbf{V})$, define define $\mathsf{ReflInfo}\langle N,\Gamma\rangle = \bigcap\{\mathsf{Info}_{\mathbf{D}}\langle M\rangle \mid M \in \mathsf{LDB}(\mathbf{D})$ and $\gamma(M) = N\}$. Thus, $\mathsf{ReflInfo}\langle N,\Gamma\rangle$ is the information common to every $M \in \mathsf{LDB}(\mathbf{D})$ which maps to N under γ. It is crucial to observe that, in general, $\mathsf{ReflInfo}\langle N,\Gamma\rangle$ is strictly larger than $\mathsf{Info}_{\mathbf{D}}\langle\bigcap\{M \in \mathsf{LDB}(\mathbf{D}) \mid \gamma(M) = N\}\rangle$. Although there are many details which must be addressed, the main focus of this paper is to study the conditions under which $\mathsf{ReflInfo}\langle N,\Gamma\rangle$ is a suitable representation for N.

Information content is *monotone* in the sense that for any $M_1, M_2 \in \mathsf{DB}(\mathbf{D})$, $M_1 \subseteq M_2$ iff $\mathsf{Info}_{\mathbf{D}}\langle M_1\rangle \subseteq \mathsf{Info}_{\mathbf{D}}\langle M_2\rangle$.

Definition 12 (Atomic representation of sentences in $\Upsilon^{\mathbf{D}}$). Define the *atomic representation* of $\varphi \in \Upsilon^{\mathbf{D}}$, denoted $\mathsf{AtRep}(\varphi)$, to be the set of all atoms which occur in φ as conjuncts. For example, if

$$\xi_1 = (\exists x_1)(\exists x_2)(\exists x_3)(R(\mathrm{a}_1,\mathrm{a}_2)\wedge R(\mathrm{a}_2,\mathrm{a}_3)\wedge R(x_1,x_2)\wedge R(x_2,\mathrm{a}_3)\wedge R(\mathrm{a}_3,x_3))$$

then $\mathsf{AtRep}(\xi_1) = \{R(\mathrm{a}_1,\mathrm{a}_2), R(\mathrm{a}_2,\mathrm{a}_3), R(x_1,x_2), R(x_2,\mathrm{a}_3), R(\mathrm{a}_3,x_3)\}$. Just by construction, $\mathsf{AtRep}(\varphi) \in \mathsf{AB}(\mathbf{D})$.

In the opposite direction, given $M \in \mathsf{AB}(\mathbf{D})$, define its *logical representation* $\mathsf{wfs}(M)$ to be the sentence in $\Upsilon^{\mathbf{D}}$ obtained by conjoining the atoms of M together and then prepending a sequence of existential quantifiers, one for each variable occurring in M, to this conjunction. Thus, the logical representation of $\mathsf{AtRep}(\xi_1)$ is ξ_1. Strictly speaking, $\mathsf{wfs}(M)$ is defined by this construction only up to a reordering of the quantifiers and conjuncts, but since such an ordering has no semantic significance, it will not be represented explicitly. Modulo this trivial reordering, it is clear that $\mathsf{AtRep}(-)$ and $\mathsf{wfs}(-)$ are inverse to one another, and so there is a natural bijective correspondence between $\Upsilon^{\mathbf{D}}$ and $\mathsf{AB}(\mathbf{D})$. The leads to a natural extension of the notion of information to an $M \in \mathsf{AB}(\mathbf{D})$; namely, define $\mathsf{Info}_{\mathbf{D}}\langle M\rangle = \mathsf{Closure}\langle \mathsf{wfs}(\varphi), \Upsilon^{\mathbf{D}}\rangle$.

Satisfaction of sentences in $\Upsilon^{\mathbf{D}}$ with respect to a given $M \in \mathsf{DB}(\mathbf{D})$ is very easy to test, because no universal quantifiers or negations are involved. Specifically, a sentence $\varphi \in \Upsilon^{\mathbf{D}}$ is in $\mathsf{DBMod}_{\mathbf{D}}(M)$ iff there is a substitution s of constants in Const for the variables of φ such that $s(\varphi) \subseteq M$. For example, if $\psi = (\exists x_1)(\exists x_2)(\exists x_3)(R(\mathrm{a}, \mathrm{b}, x_3) \wedge R(x_1, \mathrm{b}, \mathrm{c}) \wedge S(x_1, x_2, x_3))$ and $M = \{R(\mathrm{a}, \mathrm{b}, \mathrm{c}), R(\mathrm{a}, \mathrm{a}, \mathrm{d}), R(\mathrm{d}, \mathrm{b}, \mathrm{c}), S(\mathrm{a}, \mathrm{b}, \mathrm{c}), S(\mathrm{a}, \mathrm{b}, \mathrm{d})\}$, then using the standard notation of [Chang and Lee, 1973], $\{\mathrm{a}/x_1, \mathrm{b}/x_2, \mathrm{c}/x_3\}$ is a substitution which yields a subset of M, whence φ is true in M. This idea is more elegantly expressed using the notion of homomorphism, elaborated as follows.

Definition 13 (Homomorphisms). The idea of a homomorphism between database states was first developed and used in [Chandra and Merlin, 1977]. The definition given here follows that of [Fagin et al., 2005, Def. 2.3]. Let $M_1, M_2 \in \mathsf{AB}(\mathbf{D})$, and let $h : \mathsf{TermsOf}(M_1) \to \mathsf{TermsOf}(M_2)$. For $t = R(\tau_1, \tau_2, \ldots, \tau_k) \in M_1$, write $\bar{h}(t)$ for $R(h(\tau_1), h(\tau_2), \ldots, h(\tau_k))$. The function h is called a *homomorphism* from M_1 to M_2 if the following three conditions are satisfied.

(i) For all $\mathrm{a} \in \mathsf{ConstOf}(M_1)$, $h(\mathrm{a}) = \mathrm{a}$.

(ii) For all $x \in \mathsf{VarsOf}(M_1)$, $h(x) \in \mathsf{TermsOf}(M_2)$.

(iii) For all $t \in M_1$, $\bar{h}(t) \in M_2$.

The notation $h : M_1 \rightarrowtriangle M_2$ (with the triangular arrowhead) will be used to denote that h is a homomorphism from M_1 to M_2. In that case, $\bar{h}$ will be regarded as a function with domain M_1 and codomain M_2 (for the purposes of saying that it is injective, surjective, etc.). Thus, $\bar{h} : M_1 \to M_2$ and $h : \mathsf{TermsOf}(M_1) \to \mathsf{TermsOf}(M_2)$ are both functions, while $h : M_1 \rightarrowtriangle M_2$ is a convenient notation which mandates that these two functions exist and are well defined. Write $M_1 \sqsubseteq M_2$ just in case there is a homomorphism $h : M_1 \rightarrowtriangle M_2$.

It is easy to see that if the homomorphism h is injective, so too is $\bar{h}$. However, $\bar{h}$ need not be surjective even if h is. For example, with $M_1 = \{R(\mathrm{a})\}$ and $M_2 = \{R(\mathrm{a}), S(\mathrm{a})\}$, the identity function $\mathbf{1} : a \mapsto a$ is bijective, but $\bar{\mathbf{1}}$ is clearly not surjective. Call h *tuple surjective* if $\bar{h}$ is surjective. It is trivial, on the other hand, that the surjectivity of $\bar{h}$ requires the surjectivity of h. In particular, if h is injective and $\bar{h}$ is surjective, then both h and $\bar{h}$ must be bijective. Call such a homomorphism *fully bijective*. There is a stronger notion of equivalence which is needed, however. Call h *rigid* if for each $x \in \mathsf{VarsOf}(M_1)$, $h(x) \in \mathsf{Vars}$. Thus, a rigid homomorphism maps variables to variables and constants to themselves. If h is both rigid and fully bijective, it is called an *isomorphism*.

Combining the discussion at the end of Definition 12 with the homomorphism formalism immediately gives the following result.

Observation 14. *Let $\varphi \in \Upsilon^{\mathbf{D}}$ and let $M \in \mathsf{DB}(\mathbf{D})$. Then $M \in \mathsf{DBMod}_{\mathbf{D}}(\varphi)$ iff $\mathsf{AtRep}(\varphi) \sqsubseteq M$.* □

The extension of this result to the case in which the image is a general atombase is only slightly more complex.

Proposition 15. *Let $\varphi_1, \varphi_2 \in \Upsilon^{\mathbf{D}}$. Then $\varphi_1 \models \varphi_2$ iff $\mathsf{AtRep}(\varphi_2) \sqsubseteq \mathsf{AtRep}(\varphi_1)$.*

Proof. First of all, assume that $h : \mathsf{AtRep}(\varphi_1) \twoheadrightarrow \mathsf{AtRep}(\varphi_2)$ is a homomorphism, and let $M \in \mathsf{DBMod}_{\mathbf{D}}(\varphi_2)$. Then in view of Observation 14 above, there is a homomorphism $h_2 : \mathsf{AtRep}(\varphi_2) \twoheadrightarrow M$, whence $h_2 \circ h : \mathsf{AtRep}(\varphi_1) \rightarrow M$ is a homomorphism as well, and so invoking Observation 14 once again, it follows that $\varphi_1 \models \varphi_2$.

Conversely, assume that $\varphi_1 \models \varphi_2$. Let s be a substitution on $\mathsf{VarsOf}(\varphi_1)$ which assigns to each such variable x a distinct constant $\mathrm{a}_x \in \mathsf{Const} \setminus \mathsf{ConstOf}(\varphi_1)$. Let $M = s(\mathsf{AtRep}(\varphi_1))$. Then the function $h : \mathsf{TermsOf}(\varphi_1) \rightarrow \mathsf{TermsOf}(M)$ given on elements as the identity on $\mathsf{ConstOf}(\varphi_1)$ and by $x \mapsto s(x)$ on $\mathsf{VarsOf}(\varphi_1)$ defines a bijective homomorphism $h_1 : \mathsf{AtRep}(\varphi_1) \twoheadrightarrow M$, and so in particular $M \in \mathsf{DBMod}_{\mathbf{D}}(\varphi_1)$. Furthermore, since $\varphi_1 \models \varphi_2$, $M \in \mathsf{DBMod}_{\mathbf{D}}(\varphi_2)$ as well, and so by Observation 14 there is a homomorphism $h_2 : \mathsf{AtRep}(\varphi_2) \twoheadrightarrow M$. Now consider the composition $h_1^{-1} \circ h_2 : \mathsf{TermsOf}(\varphi_2) \rightarrow \mathsf{TermsOf}(\varphi_1)$. To establish that it defines a homomorphism, it is key to observe that $\mathsf{ConstOf}(\varphi_2) \subseteq \mathsf{ConstOf}(\varphi_1)$. Indeed, were there a constant $\mathrm{a} \in \mathsf{ConstOf}(\varphi_2) \setminus \mathsf{ConstOf}(\varphi_1)$, then no $M \in \mathsf{DBMod}_{\mathbf{D}}(\varphi_1)$ with $\mathrm{a} \notin M$ could possibly be in $\mathsf{DBMod}_{\mathbf{D}}(\varphi_2)$, yet it is easy to construct such a model. Armed with this observation, it is immediate that $h_1^{-1} \circ h_2$ maps constants to themselves, whence it defines a homomorphism $h_1^{-1} \circ h_2 : \mathsf{AtRep}(\varphi_2) \twoheadrightarrow \mathsf{AtRep}(\varphi_1)$, giving the desired result. □

Definition 16 (Structural identity and information equivalence). Let $M_1, M_2 \in \mathsf{AB}(\mathbf{D})$. Call M_1 and M_2 *structurally identical*, written $M_1 \triangleq M_2$, if there is an isomorphism $h : M_1 \twoheadrightarrow M_2$. Thus, M_1 and M_2 are structurally identical if they differ only in a renaming of variables. It is immediate that $\triangleq$ defines an equivalence relation on $\mathsf{AB}(\mathbf{D})$. $\mathsf{AB}_{\triangleq}(\mathbf{D})$ denotes the set of all equivalence classes of $\mathsf{AB}(\mathbf{D})$ under $\triangleq$. The notation $[M]$ will be used to denote the equivalence class of $M \in \mathsf{AB}(\mathbf{D})$. Similarly, call $\varphi_1, \varphi_2 \in \Upsilon^{\mathbf{D}}$ *structurally identical*, and write $\varphi_1 \triangleq \varphi_2$, precisely in the case that $\mathsf{AtRep}(\varphi_1) \triangleq \mathsf{AtRep}(\varphi_2)$. The relation $\sqsubseteq$ extends to equivalence classes in a natural way; $[M_1] \sqsubseteq [M_2]$ iff $M_1 \sqsubseteq M_2$.

Call M_1 and M_2 *information equivalent*, written $M_1 \cong M_2$, if $\mathsf{wfs}(M_1) \models\!\!\!\dashv \mathsf{wfs}(M_2)$; that is, if they define the same set of sentences in $\Upsilon^{\mathbf{D}}$. It is trivial that information equivalence implies structural identity, but the converse does not hold in general. To obtain a sort of converse, it is necessary to work with reduced sentences, as elaborated next.

Definition 17 (Minimal models and reduced sentences). Let $\varphi \in \Upsilon^{\mathbf{D}}$. Call φ *conjunct reduced* if for no proper subset M of $\mathsf{AtRep}(\varphi)$ is it the case that $\mathsf{wfs}(M)$ is logically equivalent to φ. As a simple example, $\xi_{21} = (\exists x_1)(\exists x_2)(R(x_1) \wedge R(x_2))$ is not conjunct reduced since $\mathsf{wfs}(\mathsf{AtRep}(\varphi) \setminus \{R(x_1)\}) = (\exists x_2)(R(x_2))$ is equivalent to ξ_{21}. Call $M \in \mathsf{AB}(\mathbf{D})$ *tuple minimal* if it is of the form $\mathsf{AtRep}(\varphi)$ for some conjunct reduced $\varphi \in \Upsilon^{\mathbf{D}}$.

Proposition 18 (Characterization of tuple-minimal atombases). *Let $\varphi_1, \varphi_2 \in \Upsilon^{\mathbf{D}}$ be conjunct reduced with $\varphi_1 \models\!\!\!\dashv \varphi_2$; i.e., with φ_1 and φ_2 logically equivalent. Then $\varphi_1 \triangleq \varphi_2$.*

Proof. In view of Proposition 15, there are homomorphisms $h_1 : \mathsf{AtRep}(\varphi_1) \rightarrow \mathsf{AtRep}(\varphi_2)$ and $h_2 : \mathsf{AtRep}(\varphi_2) \rightarrow \mathsf{AtRep}(\varphi_1)$. Furthermore, since φ_1 and φ_2 are conjunct reduced, these homomorphisms must be surjective, otherwise tuples could be removed while retaining logical equivalence. Thus both $h_2 \circ h_1 : \mathsf{AtRep}(\varphi_1) \rightarrow \mathsf{AtRep}(\varphi_2)$ and $h_1 \circ h_2 : \mathsf{AtRep}(\varphi_2) \rightarrow \mathsf{AtRep}(\varphi_1)$ are surjective homomorphisms. Since the sets involved are finite, this means that they must also be injective. Since the domain and codomain of each composition contains exactly the same constants, it follows that these compositions must be isomorphisms, whence $\mathsf{AtRep}(\varphi_1) \triangleq \mathsf{AtRep}(\varphi_2)$, and so $\varphi_1 \triangleq \varphi_2$, as required. □

Definition 19 (Canonical atombases). $M \in \mathsf{AB}(\mathbf{D})$ is called a *canonical atombase* if $\mathsf{wfs}(M)$ is conjunct reduced. The set of all canonical atombases for $\mathbf{D}$ is denoted $\mathsf{CanAB}(\mathbf{D})$.

Define $\mathsf{CanAB}_{\triangleq}(\mathbf{D})$ to be the set of all equivalence classes of $\mathsf{CanAB}(\mathbf{D})$ under $\triangleq$. In view of Proposition 18, two canonical atombases $M_1, M_2 \in \mathsf{CanAB}(\mathbf{D})$ are in the same equivalence class of $\mathsf{CanAB}_{\triangleq}(\mathbf{D})$ iff $\mathsf{wfs}(M_1) \models\!\!\!\dashv \mathsf{wfs}(M_2)$. Thus, for $M_1, M_2 \in \mathsf{CanAB}(\mathbf{D})$, $M_1 \cong M_2$ iff $M_1 \triangleq M_2$.

Definition 20 (Constraint satisfaction for canonical atombases). Since an $M \in \mathsf{AB}(\mathbf{D})$ is effectively a sentence in $\Upsilon^{\mathbf{D}}$ (represented as $\mathsf{wfs}(M)$), the characterization of what it means for such an M to satisfy a $\varphi \in \Upsilon^{\mathbf{D}}$ is trivial. However, database constraints are of a more general form, and so a broader characterization is required. While a direct definition in terms of sentences is possible, it is more natural to provide a definition in terms of databases.

Let $M \in \mathsf{AB}(\mathbf{D})$ and let K be a finite subset of Const. A *free completion of M which excludes K* is a pair $\langle M', h\rangle$ in which $M' \in \mathsf{DB}(\mathbf{D})$ and $h : M \rightarrow M'$

is a bijective homomorphism with the property that for every a $\in$ $\mathsf{Const}(M)$, $h(\mathrm{a}) \notin K$. In other words, in a free completion, every variable is replaced with a distinct constant which does not occur in K.

For $M \in \mathsf{AB}(\mathbf{D})$ and $\varphi \in \mathsf{WFS}(\mathbf{D})$, say that M *satisfies* φ if $M' \in \mathsf{LDB}(\mathbf{D})$ for every free completion $\langle M', h\rangle$ of M which excludes $\mathsf{ConstOf}(M) \cup \mathsf{ConstOf}(\mathsf{Constr}(\mathbf{D}))$. The phrase "every free completion" may be replaced with "some free completion" in the above definition, since the actual identities of the new constants are of no consequence. This fact will be often used in that which follows. In the case that $M \in \mathsf{DB}(\mathbf{D})$, this reduces to the usual notion of satisfaction, since M itself is its only free completion. Define $\mathsf{LAB}(\mathbf{D})$, the *legal atombases of* $\mathbf{D}$, to be the subset of $\mathsf{AB}(\mathbf{D})$ consisting of those M which satisfy every $\varphi \in \mathsf{Constr}(\mathbf{D})$.

Care must be taken with respect to the context in which this definition is applied. For example, if $\mathsf{Constr}(\mathbf{D}) = \{(\exists x)(\exists y)(R(x) \wedge R(y) \wedge (x \neq y))\}$, then $\{R(x_1), R(x_2)\}$ satisfies the constraint while the canonical, information-equivalent reduction $\{R(x_1)\}$ does not. On the other hand, if $\mathbf{D}$ consists of $R[AB]$ governed by functional dependency $B \to A$, then $\{R(x_1, \mathrm{b}), R(x_2, \mathrm{b})\}$ does not satisfy the constraint while its information-identical reduction $\{R(x_1, \mathrm{b})\}$ does. The solution is to disallow the first type of constraint and to repair the second problem by restricting models to canonical atombases. Furthermore, since constructions will sometimes yield non-canonical atombases which must be reduced, it is essential that such reductions preserve constraints which are already in place. Formally, say that $\varphi \in \mathsf{WFS}(\mathbf{D})$ is *preserved under minimization* if for every $M \in \mathsf{AB}(\mathbf{D})$ which satisfies φ, every $M' \in \mathsf{CanAB}(\mathbf{D})$ with $\mathsf{wfs}(M) \models\!\!\!\!\dashv \mathsf{wfs}(M')$ satisfies φ as well. The schema $\mathbf{D}$ is *closed under minimization* if every $\varphi \in \mathsf{Constr}(\mathbf{D})$ is preserved under minimization. The view $\Gamma = (\mathbf{V}, \gamma)$ is said to be *closed under minimization* precisely in the case that its schema $\mathbf{V}$ has that property.

Database schemata whose constraints are defined by Horn clauses are closed under minimization, as established in Proposition 22 below.

Definition 21 (Generalized Horn dependencies). The *Generalized Horn Dependencies*, or *GHDs* [Hegner, 2008a, 3.15], include virtually all traditional data dependencies which have been considered.. They are essentially the source-to-target dependencies of [Fagin et al., 2005, Def. 2.1] with the source and target schemata the same. They do not include the constrained dependencies of [Maher and Srivastava, 1996], which involve predicates, such as inequality, on variables.

Each GHD is a member of $\mathsf{WFS}(\mathbf{D})$ of the following form, in which the A_i's and B_i's are atoms.

$$(\forall x_1)\ldots(\forall x_m)((A_1 \wedge \ldots \wedge A_n) \Rightarrow ((\exists y_1)\ldots(\exists y_r)(B_1 \wedge \ldots \wedge B_s)))$$

Each x_i occurs in some A_j; no universally quantified variable occurs only in a B_j.

As in traditional dependency theory, there are two subclasses. In a *tuple-generating* dependency, or TGHD, each A_i and each B_j is in $\mathsf{Atoms}(\mathbf{D})$. In an *equality-generating* dependency, or EGHD, each $A_i \in \mathsf{Atoms}(\mathbf{D})$, but $r = 0$ (i.e., there are no existentially quantified variables), $s = 1$, and $B_1 \in \mathsf{VarEqAtoms} \cup \{\perp\}$. The set of all TGHDs (resp. EGHDs, resp. GHDs) on $\mathbf{D}$ is denoted $\mathsf{TGHD}(\mathbf{D})$ (resp. $\mathsf{EGHD}(\mathbf{D})$, resp. $\mathsf{GHD}(\mathbf{D})$).

The GHDs are more liberal than classical database dependencies in several useful ways. First of all, there is no requirement that n be greater than zero. If $n = 0$, the antecedent set is always true; thus, a sentence in $\Upsilon^{\mathbf{D}}$ is obtained; i.e., $\Upsilon^{\mathbf{D}} \subseteq \mathsf{GHD}(\mathbf{D})$. As a specific example of such a constraint, consider $(\exists y_1)(\exists y_2)(R(y_1, y_2))$, which states that the relation instance for R is always nonempty. Second, if $B_1 = \{\perp\}$, then $\varphi \in \mathsf{EGHD}(\mathbf{D})$ is called a *mutual-exclusion constraint*. Such sentences are seldom considered as database dependencies, but they do have uses in modelling. An example is the antisymmetry constraint $(\forall x_1)(\forall x_2)((R(x_1, x_2) \wedge R(x_2, x_1)) \Rightarrow \perp)$. Finally, constant symbols are allowed in a GHD. For example, $(\forall x_1)(\forall x_2)(R(x_1, x_2) \Rightarrow S(x_1, x_2, \nu))$ might assert that for every R-tuple, a corresponding S-tuple, padded with the constant ν in the third position (representing a null value, for example), is required.

The problems of satisfaction and inference on Horn-sentence constraints has been studied extensively, particularly in the context of an inference procedure known as the *chase*. Originally developed over thirty years ago for the study of basic database dependencies [Maier et al., 1979], it has recently seen renewed interest in many areas of database theory, including in particular data exchange [Fagin et al., 2005]. Because comprehensive and up-to-date presentations already exist [Meier et al., 2009], the details will not be repeated here.

Given a GHD φ of the above form, $\{A_1, \ldots, A_n\}$ is called the *antecedent set*, denoted $\mathsf{Antc}(\varphi)$, while $\{B_1, \ldots, B_m\}$ is called the *consequent set*, denoted $\mathsf{Cnsq}(\varphi)$. There is a convenient way to represent satisfaction of a GHD in terms of homomorphisms [Meier et al., 2009]. For φ a TGHD, $M \in \mathsf{AB}(\mathbf{D})$ satisfies φ iff every homomorphism $h : \mathsf{Antc}(\varphi) \rightarrow M$ extends to a homomorphism $h' : \mathsf{Antc}(\varphi) \cup \mathsf{Cnsq}(\varphi) \rightarrow M$ which agrees with $\bar{h}$ on $\mathsf{Antc}(\varphi)$. For φ an EGHD, $M \in \mathsf{AB}(\mathbf{D})$ satisfies φ iff every homomorphism $h : \mathsf{Antc}(\varphi) \rightarrow M$ respects the equality embodied in $\mathsf{Cnsq}(\varphi)$; i.e., if $\mathsf{Cnsq}(\varphi) = \{\tau = \tau'\}$, then $h(\tau) = h(\tau')$. (Special case: the "equality" $\perp$ can never be satisfied, so an exclusion dependency fails on φ if there is a homomorphism $h : \mathsf{Antc}(\varphi) \rightarrow M$.)

Proposition 22 (Minimization preserves GHDs). *If* $\varphi \in \mathsf{GHD}(\mathbf{D})$*, then* φ *is preserved under minimization. Thus, if* $\mathsf{Constr}(\mathbf{D}) \subseteq \mathsf{GHD}(\mathbf{D})$*, then* $\mathbf{D}$ *is closed under minimization.*

Proof. Let $M \in \mathsf{LAB}(\mathbf{D})$, and let $M' \in \mathsf{CanAB}(\mathbf{D})$ with $\mathsf{wfs}(M) \models\!\mid \mathsf{wfs}(M')$. Without loss of generality, assume that $M' \subseteq M$; i.e., that M' is obtained from M by removing tuples, and let $g : M' \twoheadrightarrow M$ be the natural inclusion homomorphism. Since $\mathsf{Info}_{\mathbf{D}}\langle M\rangle = \mathsf{Info}_{\mathbf{D}}\langle M'\rangle$, there is also a homomorphism $h : M \twoheadrightarrow M'$. Furthermore, $h \circ g : M' \twoheadrightarrow M'$ must be an isomorphism, since M' is reduced. Without loss of generality, h may be chosen so that $h \circ g$ is the identity. Now, let $\varphi \in \mathsf{TGHD}(\mathbf{D})$, and assume that M satisfies φ. Let $h_A : \mathsf{Antc}(\varphi) \twoheadrightarrow M'$; then $g \circ h_A : \mathsf{Antc}(\varphi) \twoheadrightarrow M$, and so extends to a homomorphism $h_{AC} : \mathsf{Antc}(\varphi) \cup \mathsf{Cnsq}(\varphi) \twoheadrightarrow M$. Then $h \circ g \circ h_A = h_A : \mathsf{Antc}(\varphi) \twoheadrightarrow M'$ extends to $h \circ h_{AC} : \mathsf{Antc}(\varphi) \cup \mathsf{Cnsq}(\varphi) \twoheadrightarrow M'$, thus establishing that M' satisfies φ.

For $\varphi \in \mathsf{EGHD}(\mathbf{D})$, the proof is trivial, since if M satisfies φ, then any $M' \subseteq M$ also satisfies φ. □

Definition 23 (Canonical databases with constraints). Let $\mathsf{CanLAB}(\mathbf{D})$ denote the set of all $M \in \mathsf{CanAB}(\mathbf{D})$ which satisfy every $\varphi \in \mathsf{Constr}(\mathbf{D})$. Thus, $\mathsf{CanLAB}(\mathbf{D}) = \mathsf{CanAB}(\mathbf{D}) \cap \mathsf{LAB}(\mathbf{D})$. Define $\mathsf{CanLAB}_{\triangleq}(\mathbf{D})$ to be the subset of $\mathsf{CanAB}_{\triangleq}(\mathbf{D})$ consisting of just those equivalence classes containing members of $\mathsf{CanLAB}(\mathbf{D})$. If $\mathbf{D}$ is closed under minimization, then by Proposition 22, for every $M \in \mathsf{LAB}(\mathbf{D})$, there is a block $[M'] \in \mathsf{CanLAB}_{\triangleq}(\mathbf{D})$ with $M \models\!\!\!\!\dashv M'$.

A given $M \in \mathsf{DB}(\mathbf{D})$ is equivalent only to itself under $\triangleq$, so each complete database M defines an equivalence class of $\mathsf{CanAB}_{\triangleq}(\mathbf{D})$ containing only itself. As a slight abuse of notation, each $M \in \mathsf{DB}(\mathbf{D})$ will also be regarded as a member of $\mathsf{AB}_{\triangleq}(\mathbf{D})$ and $\mathsf{CanAB}_{\triangleq}(\mathbf{D})$, and each $M \in \mathsf{LDB}(\mathbf{D})$ will be regarded as a member of $\mathsf{CanLAB}_{\triangleq}(\mathbf{D})$ as well.

Definition 24 (Extended information). In the constructions for embeddable views of the next section, it will be crucial to regard a given $G \in \mathsf{AB}(\mathbf{D})$ as a "skeleton" for a $G' \in \mathsf{CanLAB}(\mathbf{D})$ in the sense that G' is the least extension of G to $\mathsf{CanLAB}(\mathbf{D})$. To this end, define the *extended information* of G with respect to $\mathbf{D}$ [Hegner, 2008a, 3.11(b)] to be $\mathsf{XInfo}_{\mathbf{D}}\langle G\rangle = \{\varphi \in \Upsilon^{\mathbf{D}} \mid \mathsf{Constr}(\mathbf{D}) \cup \mathsf{Info}_{\mathbf{D}}\langle G\rangle \models \varphi\}$ if $\mathsf{Constr}(\mathbf{D}) \cup \mathsf{Info}_{\mathbf{D}}\langle G\rangle$ is satisfiable, and $\{\bot\}$ if it is not satisfiable. Thus, the extended information of G identifies the information which must hold in every $G' \in \mathsf{LAB}(\mathbf{D})$ with $\mathsf{Info}_{\mathbf{D}}\langle G\rangle \subseteq \mathsf{Info}_{\mathbf{D}}\langle G'\rangle$. If G is to have such a least extension, it is clear that it must be given by an $M \in \mathsf{CanLAB}(\mathbf{D})$ with $\mathsf{wfs}(M) \models\!\!\!\!\dashv \mathsf{XInfo}_{\mathbf{D}}\langle G\rangle$.

Even if it is not $\{\bot\}$, there is no guarantee that $\mathsf{XInfo}_{\mathbf{D}}\langle G\rangle$ represents the information of some $G' \in \mathsf{AB}(\mathbf{D})$, because there is no guarantee that $\mathsf{XInfo}_{\mathbf{D}}\langle G\rangle$ is finitely generated. If it is not finitely generated, then a database which would recapture its information would necessarily be infinite. Call $G \in \mathsf{AB}(\mathbf{D})$ *finitary* in $\mathbf{D}$ if $\mathsf{XInfo}_{\mathbf{D}}\langle G\rangle \neq \{\bot\}$ and is finitely generated in $\Upsilon^{\mathbf{D}}$. Say that $\mathbf{D}$ has the *finite-generation property* if every $G \in \mathsf{AB}(\mathbf{D})$ for which $\mathsf{XInfo}_{\mathbf{D}}\langle G\rangle \neq \{\bot\}$ is finitary. If G is finitary, $\mathsf{XInfo}_{\mathbf{D}}\langle G\rangle$ must have a basis consisting of a single element $\psi_G \in \Upsilon^{\mathbf{D}}$ (just conjoin the elements of the finite cover). In this case, a *semantic extension* in $\mathbf{D}$ of G is any $G' \in \mathsf{CanLAB}(\mathbf{D})$ with $\mathsf{Info}_{\mathbf{D}}\langle G'\rangle \models\!\!\!\!\dashv \psi$. All such extensions are structurally identical, and so it is natural to view the collection as a member of $\mathsf{CanAB}_{\triangleq}(\mathbf{D})$. In the case that it exists, let $\mathsf{SemExt}_{\mathbf{D}}\langle G\rangle \in \mathsf{CanAB}_{\triangleq}(\mathbf{D})$ denote the semantic extension of G in $\mathbf{D}$.

$\mathsf{SemExt}_{\mathbf{D}}\langle G\rangle$ need not be a member of $\mathsf{CanLAB}_{\triangleq}(\mathbf{D})$. For example, the semantic completion of the state $M_{53} = \{R(\mathrm{a}_0)\} \in \mathsf{DB}(\mathbf{E}_5)$ of the schema $\mathbf{E}_5$ of Examples 37 is its own semantic extension but is not in $\mathsf{LDB}(\mathbf{E}_5)$. Say that $\mathbf{D}$ has the *semantic extension property* if it has the finite-generation property, and for each $G \in \mathsf{AB}(\mathbf{D})$, $\mathsf{SemExt}_{\mathbf{D}}\langle G\rangle \in \mathsf{CanLAB}_{\triangleq}(\mathbf{D})$. Fortunately, a schema constrained by GHDs has the semantic extension property, provided it has the finite-generation property. Indeed, if the chase procedure terminates on a $G \in \mathsf{AB}(\mathbf{D})$, then it provides a $G \in \mathsf{LAB}(\mathbf{D})$ whose reduction is a semantic extension.[1] In [Meier et al., 2009], this problem of termination is studied extensively, and many conditions which guarantee termination are identified. An information-oriented presentation of the same ideas is given in [Hegner, 2008a, Thm. 3.20]. In any case, the following holds.

Proposition 25. *If* $\mathsf{Constr}(\mathbf{D})$ *is a finite set of GHDs which has the finite-generation property, then* $\mathbf{D}$ *has the semantic-extension property as well.* □

4 Embeddable Views

Using the tools developed in the previous section, the central idea of embedding a view into the main schema is developed in this section. The overall roadmap of the strategy is depicted in Fig. 1. Although the concrete examples all lie within the domain of schemata constrained by GHDs and views defined by conjunctive queries, it nevertheless seems appropriate to present the definitions independently of these concrete contexts. The two main steps of this general approach are shown on the top line. In the first step, the usual notion of database mapping is extended to canonical atombases. In the second step, the information embodied in the view state is lifted back to the main schema, in a manner analogous to the semantic morphisms of [Hegner, 2008c]. The steps shown in dashed lines represent the constructions for the concrete case of views defined by conjunctive queries and schemata constrained by GHDs. In the first step, the definition of the view in the relational calculus, as a query, is used to define the information in the main schema necessary to represent the state of the view. In the second step. this information is extended semantically, to satisfy the GHDs of the main schema. For reasons of compactness of presentation, all examples have been placed at the end of the section in Examples 37. Nevertheless, to tie the

[1] The chase procedure, as presented in [Fagin et al., 2005] and [Meier et al., 2009], does not include all of the extensions incorporated in GHDs, such as information sentences and mutual-exclusion constraints. However, the modifications necessary to incorporate them into the algorithms of those papers are very straightforward, and will not be elaborated here.

$$\begin{tikzcd}[column sep=large]
\begin{array}{c}\text{Main state}\\ {[M]}\\ \mathsf{CanLAB}_{\triangleq}(\mathbf{D})\end{array} \arrow[r, "\substack{\text{View morphism}\\ \gamma_{\mathsf{CanLAB}_{\triangleq}}}"] & \begin{array}{c}\text{View state}\\ {[N]}\\ \mathsf{CanLAB}_{\triangleq}(\mathbf{V})\end{array} \arrow[r, "\substack{\text{Semantic}\\ \text{embedding}\\ \gamma_{\mathsf{SmEmb}}}"] \arrow[dr, dashed, bend right, "\substack{\text{Syntactic}\\ \text{representation}\\ \mathsf{SynRep}\langle -, \Gamma \rangle}"'] & \begin{array}{c}\text{Representation}\\ \text{of } [N] \text{ in } \mathbf{D}:\\ \mathsf{SemRep}\langle [N], \Gamma \rangle\\ \mathsf{CanLAB}_{\triangleq}(\mathbf{D})\end{array} \\
& & \mathsf{AB}_{\triangleq}(\mathbf{D}) \arrow[u, dashed, "\substack{\text{Semantic}\\ \text{extension}\\ \mathsf{SemExt}_{\mathbf{D}}\langle - \rangle}"']
\end{tikzcd}$$

Figure 1: Internal representation for the view $\Gamma = (\mathbf{V}, \gamma)$

abstract definitions to concrete examples, the reader may wish to look ahead to these examples as this section is studied.

Notation 26. Throughout this section, unless explicitly stated to the contrary, take $\mathbf{D}$ to be a constrained database schema which is closed under minimization. Also, $\Gamma = (\mathbf{V}, \gamma)$ will be taken to be an LDB-view of $\mathbf{D}$; further conditions will be introduced in Notation 30.

Lemma 27. *If Γ is an external LDB-view of $\mathbf{D}$, then for every $M \in \mathsf{CanLAB}(\mathbf{D})$, $\gamma_{\mathsf{AB}}(M) \in \mathsf{LAB}(\mathbf{V})$. Furthermore, if $M' \in \mathsf{CanLAB}(\mathbf{D})$ with $M \triangleq M'$, then $\gamma_{\mathsf{AB}}(M) \triangleq \gamma_{\mathsf{AB}}(M')$ as well.*

Proof. Let $K = \mathsf{ConstOf}(M) \cup \mathsf{ConstOf}(\mathsf{Constr}(\mathbf{D})) \cup \mathsf{ConstOf}(\gamma) \cup \mathsf{ConstOf}(\mathsf{Constr}(\mathbf{V}))$, and let $\langle M', h \rangle$ be a free completion of M which excludes K. Then $\gamma_{\mathsf{AB}}(M') \in \mathsf{LDB}(\mathbf{V})$, just by the definition of LDB-view. Let $K' = \{h(x) \mid x \in \mathsf{VarsOf}(M)\}$, and let $N \in \mathsf{AB}(\mathbf{V})$ be obtained by replacing each $\mathrm{a} \in K'$ with $h^{-1}(\mathrm{a})$ in $\gamma_{\mathsf{AB}}(M')$. Thus, the new constants introduced by h are replaced by the corresponding variables. It is easy to see that $\langle \gamma_{\mathsf{AB}}(M'), h' \rangle$ is a free completion of N, where h' is the identity on $\mathsf{ConstOf}(N)$ and $h'(x) = h(x)$ for $x \in \mathsf{VarsOf}(N)$. Hence $N \in \mathsf{LAB}(\mathbf{D})$. That $\gamma_{\mathsf{AB}}(M) = N$ is immediate from the construction.

The structural equivalence of $\gamma_{\mathsf{AB}}(M)$ and $\gamma_{\mathsf{AB}}(M')$, given that property for M and M', follows directly from the above argument. $\square$

Definition 28 (External CanLAB-views). In the context of Lemma 27, there is no guarantee that $\gamma_{\mathsf{AB}}(M)$ will be in $\mathsf{CanLAB}(\mathbf{V})$, as the example surrounding N_{22} of Examples 37 illustrates. However, provided that Γ is closed under minimization, $\gamma_{\mathsf{AB}}(M)$ can be reduced to such a state. More formally, if Γ is closed under minimization and $\gamma_{\mathsf{AB}}(M) \in \mathsf{LAB}(\mathbf{V})$ for each $M \in \mathsf{CanLAB}(\mathbf{D})$, call γ a CanLAB-*morphism* and Γ a CanLAB-*view*. $\gamma_{\mathsf{AB}}(M)$ need not be in $\mathsf{CanLAB}(\mathbf{V})$, but it may always be reduced to such a database, since $\mathbf{V}$ is closed under minimization. In that case, let $\gamma_{\mathsf{CanLAB}_{\underline{\triangle}}} : \mathsf{CanLAB}_{\underline{\triangle}}(\mathbf{D}) \to \mathsf{CanLAB}_{\underline{\triangle}}(\mathbf{V})$ denote the function which sends $[M] \in \mathsf{CanLAB}_{\underline{\triangle}}(\mathbf{D})$ to the the equivalence class of all $N \in \mathsf{CanLAB}(\mathbf{V})$ which are information equivalent to $\gamma_{\mathsf{AB}}(M)$.

Definition 29 (Information-monotonic views). The view Γ is *information monotonic* if for any $M_1, M_2 \in \mathsf{AB}(\mathbf{D})$ with $\mathsf{Info}_{\mathbf{D}}\langle M_1\rangle \subseteq \mathsf{Info}_{\mathbf{D}}\langle M_2\rangle$, $\mathsf{Info}_{\mathbf{V}}\langle \gamma_{\mathsf{AB}}(M_1)\rangle \subseteq \mathsf{Info}_{\mathbf{V}}\langle \gamma_{\mathsf{AB}}(M_2)\rangle$. As long as none of the defining formulas of the from γ^R involves any negation, then Γ is information monotonic. In particular, all views of class $\exists\wedge+$ are information monotonic.

Notation 30. For the rest of this section, unless specifically stated to the contrary, let $\Gamma = (\mathbf{V}, \gamma)$ be a variable normalized and information monotonic external view of $\mathbf{D}$ which is closed under minimization. In light of Lemma 27, this implies that Γ must be a CanLAB-view.

Definition 31 (Internal representation of an external view). A representation of a view state $[N] \in \mathsf{CanLAB}_{\underline{\triangle}}(\mathbf{V})$ is a state $[G] \in \mathsf{CanLAB}_{\underline{\triangle}}(\mathbf{D})$ of the main schema which embodies the information $\mathsf{ReflInfo}\langle N, \Gamma\rangle$ common to all states of $\mathbf{D}$ which map to $[N]$. This may be recaptured succinctly by requiring that it be initial amongst all states which map to $[N]$. Formally, a *semantic representation* of $[N] \in \mathsf{CanLAB}_{\underline{\triangle}}(\mathbf{V})$ is a $[G] \in \mathsf{CanLAB}_{\underline{\triangle}}(\mathbf{D})$ with the property that $\gamma_{\mathsf{CanLAB}_{\underline{\triangle}}}([G]) = [N]$, and for any $[M] \in \mathsf{CanLAB}_{\underline{\triangle}}(\mathbf{D})$ with $\gamma_{\mathsf{CanLAB}_{\underline{\triangle}}}([M]) = [N]$, $[G] \sqsubseteq [M]$. If every $[N] \in \mathsf{CanLAB}_{\underline{\triangle}}(\mathbf{V})$ admits a semantic representation, then Γ is called *semantically embeddable* (in $\mathbf{D}$). It is immediate that any two semantic representations of $[N]$ are structurally identical, since they must contain the same information. If $[N] \in \mathsf{CanLAB}_{\underline{\triangle}}(\mathbf{V})$ is semantically embeddable, define the *semantic representation set* of N along Γ, denoted $\mathsf{SemRep}\langle [N], \Gamma\rangle$, as the equivalence class in $\mathsf{CanLAB}_{\underline{\triangle}}(\mathbf{D})$ containing all semantic representations of $[N]$.

If Γ is semantically embeddable, the function which sends each $[N] \in \mathsf{CanLAB}_{\underline{\triangle}}(\mathbf{V})$ to $\mathsf{SemRep}\langle [N], \Gamma\rangle$ is called the *semantic embedding function* for Γ, and is denoted $\gamma_{\mathsf{SmEmb}} : \mathsf{CanLAB}_{\underline{\triangle}}(\mathbf{V}) \to \mathsf{CanLAB}_{\underline{\triangle}}(\mathbf{D})$. The composition $\gamma_{\mathsf{SmEmb}} \circ \gamma_{\mathsf{CanLAB}_{\underline{\triangle}}}$ is called the *internal representation function* for Γ, and is denoted $\overrightarrow{\gamma} : \mathsf{CanLAB}_{\underline{\triangle}}(\mathbf{D}) \to \mathsf{CanLAB}_{\underline{\triangle}}(\mathbf{D})$. It represents the top line of Fig. 1.

The definition of $\overrightarrow{\gamma}$ provides the desired representation of Γ entirely within the schema $\mathbf{D}$. The definition is, however, completely abstract. The next step is to develop an explicit representation for $\overrightarrow{\gamma}$ based upon the view interpretation formulas and the constraints of $\mathbf{D}$, as sketched by the dashed lines of Fig. 1.

Definition 32 (Syntactic representation and lift). $[G] \in \mathsf{AB}_{\underline{\triangle}}(\mathbf{D})$ is a *syntactic representation* of $[N] \in \mathsf{CanLAB}_{\underline{\triangle}}(\mathbf{V})$ if for every $[M] \in \mathsf{CanLAB}_{\underline{\triangle}}(\mathbf{D})$, $[N] \sqsubseteq \gamma_{\mathsf{CanLAB}_{\underline{\triangle}}}([M])$ iff $[G] \sqsubseteq [M]$. If every $[N] \in \mathsf{CanLAB}_{\underline{\triangle}}(\mathbf{V})$ admits a syntactic representation, then Γ is called *syntactically embeddable* (in $\mathbf{D}$). Since

Γ is required to be information monotonic, every semantic representation is a syntactic representation, although the converse fails to hold, as will be seen in Examples 37 below.

For a view of class $\exists\wedge+$, it is possible to construct explicitly a syntactic representation for $[N] \in \mathsf{CanLAB}_{\triangleq}(\mathbf{V})$. Choose any representative $N \in [N]$, and define the *syntactic lift* of N along Γ to be $\mathsf{SynLift}\langle N, \Gamma\rangle = \bigcup\{\mathsf{AtRep}(\mathsf{Substf}\langle\gamma, t\rangle) \mid t \in N\}$. Thus $\mathsf{SynLift}\langle N, \Gamma\rangle \in \mathsf{AB}(\mathbf{D})$. Define the *canonical syntactic representation* of N in Γ, denoted $\mathsf{SynRep}\langle N, \Gamma\rangle$, to be the equivalence class of $\mathsf{SynLift}\langle N, \Gamma\rangle$ in $\mathsf{AB}_{\triangleq}(\mathbf{D})$. This construction is independent of the particular choice of N from in $[N]$ and so the notation $\mathsf{SynRep}\langle [N], \Gamma\rangle$ will be employed.

The following observation is immediate from the construction.

Observation 33. *If Γ is of class $\exists\wedge+$, then for every $[N] \in \mathsf{CanLAB}_{\triangleq}(\mathbf{V})$, $\mathsf{SynRep}\langle [N], \Gamma\rangle$ is a syntactic representation of $[N]$ along Γ. In particular, Γ is syntactically embeddable.* $\Box$

The construction which formalizes the dotted lower path of Fig. 1 is developed explicitly in the next three items.

Lemma 34. *If $\mathbf{D}$ has the semantic extension property, then for any $G \in \mathsf{AB}(\mathbf{D})$ and $G' \in \mathsf{CanLAB}(\mathbf{D})$, $G \sqsubseteq G'$ iff $\mathsf{SemExt}_{\mathbf{D}}\langle G\rangle \sqsubseteq [G']$.*

Proof. Suppose that $G \sqsubseteq G'$. Then it must be the case that $\mathsf{Info}_{\mathbf{D}}\langle G\rangle \subseteq \mathsf{Info}_{\mathbf{D}}\langle G'\rangle$. Since $\mathsf{XInfo}_{\mathbf{D}}\langle G\rangle$ is the least information which any $G'' \in \mathsf{LAB}(\mathbf{D})$ with $\mathsf{Info}_{\mathbf{D}}\langle G\rangle \subseteq \mathsf{Info}_{\mathbf{D}}\langle G''\rangle$ may have, it follows that $\mathsf{XInfo}_{\mathbf{D}}\langle G\rangle \subseteq \mathsf{Info}_{\mathbf{D}}\langle G'\rangle$, whence $\mathsf{SemExt}_{\mathbf{D}}\langle G\rangle \sqsubseteq [G']$.

An invocation of Proposition 15 establishes the converse. $\Box$

Proposition 35. *If $\mathbf{D}$ has the semantic extension property and Γ is of class $\exists\wedge+$, then Γ is semantically embeddable.*

Proof. By Observation 33, Γ is syntactically representable, with $\mathsf{SynRep}\langle [N], \Gamma\rangle$ a syntactic representation for a given $[N] \in \mathsf{CanLAB}_{\triangleq}(\mathbf{D})$. The result now follows from Lemma 34. $\Box$

Theorem 36 (Explicit semantic representation). *If $\mathsf{Constr}(\mathbf{D})$ is a set of GHDs with the finite-generation property and Γ is of class $\exists\wedge+$, then Γ is semantically embeddable, and for any $[M] \in \mathsf{CanLAB}_{\triangleq}(\mathbf{D})$, $\mathsf{SemRep}\langle [M], \Gamma\rangle = \mathsf{SemExt}_{\mathbf{D}}\langle \mathsf{SynRep}\langle \gamma_{\mathsf{CanLAB}_{\triangleq}}([M]), \Gamma\rangle\rangle$.*

Proof. The proof follows from Proposition 35 and Proposition 25. $\Box$

Examples 37. In order to clarify the ideas of the syntactic and semantic representation, some examples are in order. First, let $\mathbf{E}_2$ be the relational schema with the single relation symbol $R[AB]$ and no constraints. Let $\Pi_A^{\mathbf{E}_2} = (\mathbf{W}_{21}, \pi_A^{\mathbf{E}_2})$

be the view whose schema $\mathbf{W}_{21}$ has the single relational symbol $R_A[A]$ and whose view morphism $\pi_A^{\mathbf{E}_2}$ is the projection of $R[AB]$ onto $R_A[A]$. Let $N_{21} = \{R_A(\mathrm{a}_0), R_A(\mathrm{a}_1)\} \in \mathsf{LDB}(\mathbf{W}_{21})$. Then a member of $\mathsf{SynRep}\langle [N_{21}], \mathit{\Pi}_A^{\mathbf{E}_2}\rangle$ has the form $\{R(\mathrm{a}_0, x_0), R(\mathrm{a}_1, x_1)\}$, and all such members differ only in renaming of x_0 and x_1. In this example, $\mathsf{SynRep}\langle [N], \mathit{\Pi}_A^{\mathbf{E}_2}\rangle$ and $\mathsf{SemRep}\langle [N], \mathit{\Pi}_A^{\mathbf{E}_2}\rangle$ are identical for all $[N] \in \mathsf{CanLAB}(\mathbf{W}_{21})$. The internal representation function $\overrightarrow{\pi_A^{\mathbf{E}_2}}$ sends, for example, $[\{R(\mathrm{a}_0, \mathrm{b}_0), R(\mathrm{a}_1, \mathrm{b}_1)\}]$ to $[\{R(\mathrm{a}_0, x_0), R(\mathrm{a}_1, x_1)\}]$; that is, it replaces the constants b_0 and b_1 with distinct variables.

Next, consider a setting which is identical to the above example, save that the main schema now has an additional relation symbol $S[AB]$ whose instance is forced to be identical to that of R. More precisely, let $\mathbf{E}_3$ have two relation symbols $R[AB]$ and $S[AB]$, constrained by $(\forall x)(\forall y)(R(x,y) \Leftrightarrow S(x,y))$. The view $\mathit{\Pi}_A^{\mathbf{E}_3} = (\mathbf{W}_{31}, \pi_{R[A]}^{\mathbf{E}_3})$ has $\mathbf{W}_{31} = \mathbf{W}_{21}$, with $\pi_{R[A]}^{\mathbf{E}_3}$ the projection of $R[AB]$ onto $R_A[A]$. Syntactic representations are not unique in this context. Indeed, an alternative syntactic representation replaces R with S. Thus, $[\{S(\mathrm{a}_0, x_0), S(\mathrm{a}_1, x_1)\}]$ is also a syntactic representation of $[N_{21}]$, as is $[\{R(\mathrm{a}_0, x_0), S(\mathrm{a}_1, x_1)\}]$. On the other hand, a minimal semantic representation must now include both R and S, so such a representation for $[N_{21}]$ would be $[M_{3s}] = [\{R(\mathrm{a}_0, x_0), R(\mathrm{a}_1, x_1), S(\mathrm{a}_0, x_0), S(\mathrm{a}_1, x_1)\}]$. Note that $[M_{3s}]$ is a syntactic representation as well.

Now return to $\mathbf{E}_2$ above, but let $\mathit{\Pi}_{A+B}^{\mathbf{E}_2} = (\mathbf{W}_{32}, \pi_{A+B}^{\mathbf{E}_2})$ be the view whose schema $\mathbf{W}_{32}$ contains two relation symbols $R_A[A]$ and $R_B[B]$, and whose view morphism $\pi_{A+B}^{\mathbf{E}_2}$ includes the projections of $R[AB]$ onto $R_A[A]$ and onto $R_B[B]$. Thus $(\pi_{A+B}^{\mathbf{E}_2})_{\mathsf{LDB}} : \{R(\mathrm{a}, \mathrm{b})\} \mapsto \{R_A(\mathrm{a}), R_B(\mathrm{b})\}$, with the obvious extension to larger sets. Let $[N_{22}] = [\{R_A(\mathrm{a}), R_B(\mathrm{b})\}]$; then $\mathsf{SemRep}\langle [N_{22}], \mathit{\Gamma}\rangle = [\{R(\mathrm{a}, x), R(x, \mathrm{b})\}]$. Let $M_{22} = \{R(\mathrm{a}, x), R(x, \mathrm{b})\}$; then $(\pi_{A+B}^{\mathbf{E}_2})_{\mathsf{AB}}(M_{22}) = \{R_A(\mathrm{a}), R_B(\mathrm{b}), R_A(x), R_B(x)\}$ and not just N_{22}. In the terminology of [Hegner, 2008a, 4.8], *orphan tuples* have arisen. Orphan tuples may be avoided by requiring that the view *reflect deletions* [Hegner, 2008a, 4.9-4.11]; that is, by requiring that each deletion update to the view be representable as a deletion update in the main schema. However, in the current context, there is no need to enforce such a condition. Orphan tuples add no information and so are harmless in terms of semantics; they are simply removed in the process of minimization. This example does nevertheless illustrate that it cannot be expected that the image of a canonical legal database be itself canonical.

The examples considered so far do not involve any nontrivial constraints. However, constraints play an important role. To illustrate, let $\mathbf{E}_4$ be the schema with two relational symbols $R[AB]$ and $S[BC]$, subject to the inclusion dependency $R[B] \subseteq S[B]$. Let $\mathit{\Pi}_A^{\mathbf{E}_4} = (\mathbf{W}_{41}, \pi_A^{\mathbf{E}_4})$ be a view which is similar to $\mathit{\Pi}_A^{\mathbf{E}_2}$. $\mathbf{W}_{41}$ has the single relation symbol $R_A[A]$, and $\pi_A^{\mathbf{E}_4}$ projects $R[AB]$ onto $R_A[A]$ and ignores $S[BC]$. Let $N_{41} = \{R_A(\mathrm{a}_0)\} \in \mathsf{LDB}(\mathbf{W}_{41})$. Then $\mathsf{SemRep}\langle N_{41}, \mathit{\Pi}_A^{\mathbf{E}_4}\rangle$ is the equivalence class containing states of the form $M_{41} =$

$\{R(\mathrm{a}_0, x_0), S(x_0, y_0)\}$. In particular, the presence of the tuple $S(x_0, y_0)$ is mandated by the inclusion dependency. On the other hand, a state of the form $M_{42} = \{R(\mathrm{a}_0, x_0)\}$ is a syntactic representation of N_{41} but not a semantic one.

Not all constraints are suitable for a theory of representations. For example, let $\mathbf{E}_5$ have three unary relation symbols $R[A]$, $S[A]$, and $T[A]$, with the constraint $(\forall x)(R(x) \Rightarrow (S(x) \vee T(x))$. Let $\mathit{\Pi}_R^{\mathbf{E}_5} = (\mathbf{W}_{51}, \pi_R^{\mathbf{E}_5})$ be the view whose schema $\mathbf{W}_{51}$ contains the single relation symbol $R'[A]$ and whose view morphism $\pi_R^{\mathbf{E}_5}$ preserves $R[A]$ (as $R'[A]$) but ignores $S[A]$ and $T[A]$ completely. The view state $N_{51} = \{R'(\mathrm{a}_0)\}$ has the syntactic representation $M_{51} = \{R(\mathrm{a}_0)\}$ in $\mathbf{E}_5$. However, $\mathsf{SemRep}\langle N_{51}, \mathit{\Pi}_R^{\mathbf{E}_5}\rangle$ does not exist, since $\mathsf{SemExt}_{\mathbf{D}}\langle M_{51}\rangle = M_{51} \notin \mathsf{CanLAB}_{\triangleq}(\mathbf{D})$. This illustrates the importance of the semantic extension property and the consequent importance of GHDs.

Finally, it is important to note that it is not only the constraints on the main schema but also the nature of the view morphism which is important in determining whether or not the view is canonically embeddable. Let $\mathbf{E}_6$ have the two unary relation symbols $R[A]$ and $S[A]$ with no additional constraints, and define the view $\mathit{\Pi}_{R\vee S}^{\mathbf{E}_6} = (\mathbf{W}_{61}, \omega_{61}^{R\vee S})$ to have the schema $\mathbf{W}_{61}$ with the single unary relation symbol $T[A]$, and $(\omega_{61}^{R\vee S})^T = R(x_A) \vee S(x_A)$. Arguing as above, it is easy to see that even for a simple view state such as $N_{61} = \{T(\mathrm{a}_0)\}$, $\mathsf{SemRep}\langle N_{61}, \omega_{61}^{R\vee S}\rangle$ cannot exist. In this case, no syntactic representation exists either. This illustrates the importance of views of class $\exists\wedge+$.

5 The Partial Lattice Structure of $\mathsf{CanLAB}_{\triangleq}(\mathbf{D})$

The set $\mathsf{CanLAB}_{\triangleq}(\mathbf{D})$ of canonical legal databases of $\mathbf{D}$ admits a natural partial lattice structure, in the sense of in the sense of [Grätzer, 1978, p. 41], with join and meet corresponding to union and intersection of information, respectively. The meet is always defined, and the join is defined whenever the two states have an upper bound. The utility of this natural structure will be demonstrated in the next section, in an application to the view-update problem. In this section, these basic lattice-theoretic properties are developed.

Notation 38. Throughout this section, unless stated specifically to the contrary, take $\mathbf{D}$ to be a database schema with the semantic extension property.

Definition 39 (The join structure of $\mathsf{CanLAB}_{\triangleq}(\mathbf{D})$). Let $[M_1], [M_2] \in \mathsf{CanLAB}_{\triangleq}(\mathbf{D})$. Without loss of generality, assume that the set $\{M_1, M_2\}$ of representatives is *variable normalized*; i.e., that M_1 and M_2 have no variables in common. Define $[M_1] \sqcup [M_2] = \mathsf{SemExt}_{\mathbf{D}}\langle M_1 \cup M_2\rangle$ whenever the latter exists.

As a natural example, consider again the schema $\mathbf{E}_1$ introduced in Section 1 with the single ternary relation symbol $R[ABC]$, constrained by the join dependency $\bowtie[AB, BC]$, and the two projections $\mathit{\Pi}_{AB}^{\mathbf{E}_1} = (\mathbf{W}_{11}, \pi_{AB}^{\mathbf{E}_1})$ and

$\mathit{\Pi}^{\mathbf{E}_1}_{BC} = (\mathbf{W}_{12}, \pi^{\mathbf{E}_1}_{BC})$. Then the classical reconstruction constraint, that any state $M \in \mathsf{LDB}(\mathbf{E}_1)$ may be recovered from the join of its internalized AB projection $\overrightarrow{\pi^{\mathbf{E}_1}_{AB}}(M)$ and its internalized BC projection $\overrightarrow{\pi^{\mathbf{E}_1}_{BC}}(M)$, is recaptured by the lattice join equality: $M = \overrightarrow{\pi^{\mathbf{E}_1}_{AB}}(M) \sqcup \overrightarrow{\pi^{\mathbf{E}_1}_{BC}}(M)$. Thus, in this special but very important case, the lattice join is the same as the database join.

Observation 40. *For any $[M_1], [M_2] \in \mathsf{CanLAB}_{\triangleq}(\mathbf{D})$, whenever it exists, $[M_1] \sqcup [M_2]$ is the least upper bound of $[M_1]$ and $[M_2]$ in $\mathsf{CanLAB}_{\triangleq}(\mathbf{D})$ with respect to the ordering $\sqsubseteq$, and so defines a partial join operation on that set.* □

Definition 41 (Database product). To define the meet operation, the following product construction is central. Let $t_1, t_2 \in \mathsf{GndAtoms}(\mathbf{D})$. If t_1 and t_2 are both R-atoms for the same $R \in \mathsf{Rels}(\mathbf{D})$, define the *product* $t_1 \times t_2$ to be the attribute-by-attribute product of t_1 and t_2; i.e., the R-tuple with $t[A] = \langle t_1[A], t_2[A] \rangle$ for each $A \in \mathsf{Ar}_{\mathbf{D}}(R)$. Here $\langle t_1[A], t_2[A] \rangle$ is just an ordered pair of values. If t_1 and t_2 are atoms associated with distinct relations R_1 and R_2; i.e., t_1 is an R_1-atom and t_2 is an R_2 atom with $R_1 \neq R_2$, then $t_1 \times t_2$ is undefined. Now, for $M_1, M_2 \in \mathsf{CanAB}(\mathbf{D})$, define $M_1 \times M_2$ to be the product of all tuples in M_1 with tuples in M_2; i.e., $M_1 \times M_2 = \{t_1 \times t_2 \mid (t_1 \in M_1) \text{ and } (t_2 \in M_2) \text{ and } t_1 \times t_2 \text{ defined}\}$.

A *pair-renaming function* $\delta : \mathsf{Const} \times \mathsf{Const} \to \mathsf{Const}$ sends each pair $\langle \mathrm{b}, \mathrm{b} \rangle \in \mathsf{Const} \times \mathsf{Const}$ containing two instances of the same constant to itself., and every other pair of constants to a distinct constant. Since Const is a countable set, such a function always exists. The function δ extends naturally to products of tuples and of databases in a homomorphism-like fashion. Specifically, for $t_1, t_2 \in \mathsf{GndAtoms}(\mathbf{D})$ for the same R, define $\bar{\delta}(t_1 \times t_2)$ to be the R-tuple with $\bar{\delta}(t_1 \times t_2)[A] = \delta(\langle t_1[A], t_2[A] \rangle)$ for each $A \in \mathsf{Ar}_{\mathbf{D}}(R)$. Then, for $M_1, M_2 \in \mathsf{AB}_{\triangleq}(\mathbf{D})$, define $\bar{\delta}(M_1 \times M_2) = \{\bar{\delta}(t) \mid t \in M_1 \times M_2\}$, and define $M_1 \sqcap_\delta M_2 = \bar{\delta}(M_1 \times M_2)$.

Lemma 42. *Let $M_1, M_2 \in \mathsf{AB}_{\triangleq}(\mathbf{D})$, and let δ be any pair-renaming function. Then $\mathsf{Info}_{\mathbf{D}}\langle M_1 \sqcap_\delta M_2 \rangle = \mathsf{Info}_{\mathbf{D}}\langle M_1 \rangle \cap \mathsf{Info}_{\mathbf{D}}\langle M_2 \rangle$.*

Proof. Let $M \in \mathsf{AB}_{\triangleq}(\mathbf{D})$ with $M \sqsubseteq M_1$ and $M \sqsubseteq M_2$, so that there are homomorphisms $h_i : M \twoheadrightarrow M_i$ for $i \in \{1, 2\}$. It is easy to see that $\delta \circ \langle h_1, h_2 \rangle : x \mapsto \delta(\langle h_1(x), h_2(x) \rangle)$ defines a homomorphism $\delta \circ \langle h_1, h_2 \rangle : M \twoheadrightarrow M_1 \sqcap_\delta M_2$, whence $M \sqsubseteq M_1 \sqcap_\delta M_2$. In particular, choosing $\varphi \in \mathsf{Info}_{\mathbf{D}}\langle M_1 \rangle \cap \mathsf{Info}_{\mathbf{D}}\langle M_2 \rangle$, and fixing $M \in \mathsf{DBMod}_{\mathbf{D}}(\varphi) \cap \mathsf{CanAB}(\mathbf{D})$, it follows that $\varphi \in \mathsf{Info}_{\mathbf{D}}\langle M_1 \sqcap_\delta M_2 \rangle$; i.e., $\mathsf{Info}_{\mathbf{D}}\langle M_1 \rangle \cap \mathsf{Info}_{\mathbf{D}}\langle M_2 \rangle \subseteq \mathsf{Info}_{\mathbf{D}}\langle M_1 \sqcap_\delta M_2 \rangle$. On the other hand, for $i \in \{1, 2\}$, there is a natural homomorphism $g_i : M_1 \sqcap_\delta M_2 \to M_i$ given by $x \mapsto \pi_i(\delta^{-1}(x))$, with π_i the projection of the i^{th} element of an ordered pair. Thus $M_1 \sqcap_\delta M_2 \sqsubseteq M_i$, so $\mathsf{Info}_{\mathbf{D}}\langle M_1 \sqcap_\delta M_2 \rangle \subseteq \mathsf{Info}_{\mathbf{D}}\langle M_i \rangle$, completing the proof. □

Proposition 43 (Closure of $\mathsf{LAB}(\mathbf{D})$ under meet). *If $M_1, M_2 \in \mathsf{LAB}(\mathbf{D})$, then $M_1 \sqcap_\delta M_2 \in \mathsf{LAB}(\mathbf{D})$ for any pair-renaming function δ.*

Proof. Using Lemma 42 and the monotonicity of information (Definition 11), $\mathsf{Info}_{\mathbf{D}}\langle\mathsf{SemExt}_{\mathbf{D}}\langle M_1 \sqcap_\delta M_2\rangle\rangle \subseteq \mathsf{Info}_{\mathbf{D}}\langle\mathsf{SemExt}_{\mathbf{D}}\langle M_i\rangle\rangle = \mathsf{Info}_{\mathbf{D}}\langle M_i\rangle$ for $i \in \{1, 2\}$. Thus, applying Lemma 42 again, $\mathsf{Info}_{\mathbf{D}}\langle\mathsf{SemExt}_{\mathbf{D}}\langle M_1 \sqcap_\delta M_2\rangle\rangle \subseteq \mathsf{Info}_{\mathbf{D}}\langle M_1\rangle \cap \mathsf{Info}_{\mathbf{D}}\langle M_2\rangle = \mathsf{Info}_{\mathbf{D}}\langle M_1 \sqcap_\delta M_2\rangle$. Since $\mathsf{Info}_{\mathbf{D}}\langle M_1 \sqcap_\delta M_2\rangle \subseteq \mathsf{Info}_{\mathbf{D}}\langle\mathsf{SemExt}_{\mathbf{D}}\langle M_1 \sqcap_\delta M_2\rangle\rangle$, these two must be equal, and so $M_1 \sqcap_\delta M_2 \in \mathsf{LAB}(\mathbf{D})$, as required. □

Definition 44 (The meet structure of $\mathsf{CanLAB}_{\triangleq}(\mathbf{D})$). It is immediate from Lemma 42 that for any $M_1, M_2 \in \mathsf{AB}_{\triangleq}(\mathbf{D})$ and any pair renaming functions δ and δ', $M_1 \sqcap_\delta M_2 \triangleq M_1 \sqcap_{\delta'} M_2$. Thus, for $[M_1], [M_2], \in \mathsf{CanLAB}(\mathbf{D})$, the *meet* of $[M_1]$ and $[M_2]$, denoted $[M_1] \sqcap [M_2]$, may be (and is) defined unambiguously as the equivalence class of $\mathsf{CanAB}_{\triangleq}(\mathbf{D})$ containing a tuple-minimization of $M_1 \sqcap_\delta M_2$ for any pair-renaming function δ. By Proposition 43, this meet is also in $\mathsf{CanLAB}_{\triangleq}(\mathbf{D})$, completing the definition of the meet structure for $\mathsf{CanLAB}_{\triangleq}(\mathbf{D})$.

The information-preservation property of the meet structure is closely related to the fact that Horn sentences (and in particular Boolean conjunctive queries) are closed under product [Horn, 1951], [Fagin, 1982]. Indeed, it is not difficult to see that the product construction of Definition 41 is essentially a model-theoretic product in disguise.

6 An Application to the View-Update Problem

The problem of how best to reflect an update on a view back to the main schema has long been a subject of importance in database systems. One of the major approaches is the constant-complement strategy, introduced in [Bancilhon and Spyratos, 1981]. The idea is to decompose the main schema $\mathbf{D}$ into the view Γ_1 to be updated and a complementary view Γ_2, with the property that the state of $\mathbf{D}$ may be recovered from the combined states of Γ_1 and Γ_2. Given an update to the view Γ_1, there is at most one update to $\mathbf{D}$ which holds Γ_2 constant. The intuition behind this approach is that Γ_2 represents the part of $\mathbf{D}$ which is not part of Γ_1, and so should be fixed for the update to Γ_1. A major complication is that the reflection of the view update to the main schema, in the most general case, depends upon the choice of complement Γ_2. Despite this, it is almost always the case in practice that there is one "natural" complement, with all others appearing "contrived". In [Hegner, 2004], it is shown that by requiring the view mappings to respect the natural order structure on the databases, the reflection of a view update to the main schema is independent of the choice of complement, provided that the view update itself is decomposable into a sequence of insertions and deletions. Unfortunately, such decompositions are often not possible when working with complete databases. In this section, it is shown that by allowing atombases as intermediate results, the uniqueness result for view-update reflections may be extended beyond sequences of insertions and deletions.

Notation 45. Throughout this section, unless stated specifically to the contrary, $\mathbf{D}$ will be taken to be a database schema with the semantic-extension property, with $\mathit{\Gamma} = (\mathbf{V}, \gamma)$, $\mathit{\Gamma}_1 = (\mathbf{V}_1, \gamma_1)$, $\mathit{\Gamma}_2 = (\mathbf{V}_2, \gamma_2)$, and $\mathit{\Gamma}_3 = (\mathbf{V}_3, \gamma_3)$ semantically representable external views of $\mathbf{D}$ of class $\exists\wedge+$.

Definition 46 (Updates and reflections). The definitions given here are adapted from those of [Hegner, 2008a, 4.2]. An *update* on $\mathbf{D}$ is a pair $([M_1], [M_2]) \in \mathsf{CanLAB}_{\triangleq}(\mathbf{D}) \times \mathsf{CanLAB}_{\triangleq}(\mathbf{D})$. $[M_1]$ is the current state, and $[M_2]$ the new state. It is an *insertion* if $[M_1] \sqsubseteq [M_2]$, and a *deletion* if $[M_2] \sqsubseteq [M_1]$. It is called *complete* if $[M_i] \in \mathsf{LDB}(\mathbf{D})$ for $i \in \{1, 2\}$. To describe the situation surrounding an update request on $\mathit{\Gamma}$, it is sufficient to specify the current state $[M]$ of the main schema and the desired new state $[N]$ of the view schema $\mathbf{V}$. The current state of the view can be computed as $\gamma_{\mathsf{CanLAB}_{\triangleq}}([M])$; it is only the new state $[M']$ of the main schema (subject to $[N] = \gamma_{\mathsf{CanLAB}_{\triangleq}}([M'])$) which must be obtained from an update strategy. Formally, an *update request* from $\mathit{\Gamma}$ to $\mathbf{D}$ is a pair $([M], [N])$ in which $[M] \in \mathsf{CanLAB}_{\triangleq}(\mathbf{D})$ (the old state of the main schema) and $[N] \in \mathsf{CanLAB}_{\triangleq}(\mathbf{V})$ (the new state of the view schema). If $\gamma_{\mathsf{CanLAB}_{\triangleq}}([M]) \sqsubseteq [N]$, it is called an *insertion request*, and if $[N] \sqsubseteq \gamma_{\mathsf{CanLAB}_{\triangleq}}([M])$, it is called a *deletion request*. Collectively, insertion requests and deletion requests are termed *unidirectional update requests*. If $(M, N) \in \mathsf{LDB}(\mathbf{D}) \times \mathsf{LDB}(\mathbf{V})$, then (M, N) is called a *complete* update request. A *realization* of $([M], [N])$ along $\mathit{\Gamma}$ is an update $([M], [M'])$ on $\mathbf{D}$ with the property that $\gamma_{\mathsf{CanLAB}_{\triangleq}}([M']) = [N]$. The update $([M], [M'])$ is called a *reflection* (or *translation*) of the view update $(\gamma_{\mathsf{CanLAB}_{\triangleq}}([M]), [N])$ *with base state* $[M]$. In this work, attention will be focused upon the support of complete update requests, although updates which are not complete will be employed in a supporting rôle.

Definition 47 (Set-based complementary pairs). According to the classical definition [Bancilhon and Spyratos, 1981, Def. 4.4] , the set $\{\mathit{\Gamma}_1, \mathit{\Gamma}_2\}$ forms a *set-based complementary* pair if the decomposition mapping $(\gamma_1)_{\mathsf{LDB}} \times (\gamma_2)_{\mathsf{LDB}} : \mathsf{LDB}(\mathbf{D}) \to \mathsf{LDB}(\mathbf{V}_1) \times \mathsf{LDB}(\mathbf{V}_2)$ which is defined on elements by $M \mapsto ((\gamma_1)_{\mathsf{LDB}}(M), (\gamma_2)_{\mathsf{LDB}}(M))$ is injective. In this case, $\mathit{\Gamma}_i$ is said to be a *set-based complement* of $\mathit{\Gamma}_{3-i}$ for $i \in \{1, 2\}$. Let $(M, N) \in \mathsf{LDB}(\mathbf{D}) \times \mathsf{LDB}(\mathbf{V}_1)$ be a complete update request from $\mathit{\Gamma}_1$ to $\mathbf{D}$. There is at most one translation of this update request which keeps the state $(\gamma_2)_{\mathsf{LDB}}(M)$ constant; indeed, if a translation exists, the new state M' of $\mathbf{D}$ must be the inverse image of $(N, (\gamma_2)_{\mathsf{LDB}}(M))$ under $(\gamma_1)_{\mathsf{LDB}} \times (\gamma_2)_{\mathsf{LDB}}$, and the latter is injective just by definition of complementary pair. If it exists, it is called the *translation of* (M, N) *with constant complement* $\mathit{\Gamma}_2$, and is denoted $\mathsf{CCTransl}\langle (M, N), \mathit{\Gamma}_1; \mathit{\Gamma}_2 \rangle$.

As noted in the introduction to this section, complements are not unique and the reflection of an update request may depend upon the choice of complement; see [Hegner, 2004, Sec. 1.3] for examples. As also noted in the introduction, in [Hegner, 2004] it is shown that if order structure is imposed upon the views and the decomposition mapping is required to respect that structure, then for an *order-based* view update, that is, one which is realizable as a sequence of insertions and deletions, the translation is independent of the choice of complement.

Unfortunately, many updates are not order based. For example, let $\mathbf{E}_7$ contain the single relation symbol $R[ABC]$, constrained by the functional dependency $B \to A$. The pair of views consisting of the AB-projection $\Pi^{\mathbf{E}_7}_{AB} = (\mathbf{W}_{71}, \pi^{\mathbf{E}_7}_{AB})$ and the BC projection $\Pi^{\mathbf{E}_7}_{BC} = (\mathbf{W}_{72}, \pi^{\mathbf{E}_7}_{BC})$ forms a set-based complementary pair, and is indeed a prototypical lossless decomposition in traditional database theory. Note that $B \to A$ embeds into $\mathbf{W}_{71}$. Let $M_{71} = \{R(\mathrm{a}_1, \mathrm{b}_1, \mathrm{c}_1)\} \in \mathsf{LDB}(\mathbf{E}_7)$, and consider the update on $\Pi^{\mathbf{E}_7}_{AB}$ which replaces $(\pi^{\mathbf{E}_7}_{AB})_{\mathsf{LDB}}(M) = \{R_{AB}(\mathrm{a}_1, \mathrm{b}_1)\}$ with $\{R_{AB}(\mathrm{a}_2, \mathrm{b}_1)\}$. The translation of this update with constant complement $\Pi^{\mathbf{E}_7}_{BC}$ is (M_{71}, M_{72}) with $M_{72} = \{R(\mathrm{a}_2, \mathrm{b}_1, \mathrm{c}_1)\}$, which is not realizable as a sequence of insertions and deletions which lie in $\mathsf{LDB}(\mathbf{D})$. Thus, the theory of [Hegner, 2004] cannot address this case. However, it is realizable as the deletion $(M_{71}, [M_{73}])$ followed by the insertion $([M_{73}], M_{72})$, with $M_{73} = \{R(x, \mathrm{b}_1, \mathrm{c}_1)\} \in \mathsf{CanLAB}(\mathbf{D})$. It will now be shown how to accommodate this idea formally.

Definition 48 (Information-based complements). Call $\{\Gamma_1, \Gamma_2\}$ an *info-based complementary pair* if for all $M \in \mathsf{LDB}(\mathbf{D})$, $\mathsf{SemExt}_{\mathbf{D}}\langle \overrightarrow{\gamma_1}(M) \cup \overrightarrow{\gamma_2}(M)\rangle = M$. Thus, in an info-based complementary pair, the information of the state of the main schema is the extended information defined by the states of the two views. It is easy to see that every info-based complementary pair is also a set-based complementary pair. This is a very reasonable condition which is satisfied by common decompositions, including in particular those whose reconstruction mapping (the inverse of the decomposition mapping $(\gamma_1)_{\mathsf{LDB}} \times (\gamma_2)_{\mathsf{LDB}}$) is the join.

Unfortunately, this property does not extend uniformly to $\mathsf{CanLAB}(\mathbf{D})$. To see this, continue with the example of Definition 47 and consider the state $M_{74} = \{R(\mathrm{a}_0, x_0, \mathrm{c}_0), R(\mathrm{a}_1, x_1, \mathrm{c}_1)\} \in \mathsf{CanLAB}(\mathbf{D})$. The two projections of this state are $M_{75} = (\pi^{\mathbf{E}_7}_{AB})_{\mathsf{AB}} = \{R_{AB}(\mathrm{a}_0, x_0), R_{AB}(\mathrm{a}_1, x_1)\}$ and $M_{76} = (\pi^{\mathbf{E}_7}_{BC})_{\mathsf{AB}} = \{R_{BC}(x_0, \mathrm{c}_0), R_{BC}(x_1, \mathrm{c}_1)\}$. As each of these states is unique only up to essential identity, the variables may be renamed, so that M_{76} may be replaced with $M_{76'} = \{R_{BC}(x_2, \mathrm{c}_0), R_{BC}(x_3, \mathrm{c}_1)\}$, for example. However, M_{74} cannot be recovered from M_{75} and $M_{76'}$, and thus the decomposition is not lossless. The problem in the above example is that the variables x_1 and x_2 from M_{74} appear in both view states, where they may be independently renamed, thus deleting essential information for the reconstruction. It is for this reason that the formulation of complements and updates given above is entirely within the LDB-context;

i.e., using complete databases. Nevertheless, it is possible to use variables in a critical way in order to extend the uniqueness results to view updates which are not order based, as shown in the following theorem.

Theorem 49. *Let* $\mathbf{D}$ *have the semantic extension property, let* $\{\Gamma_1, \Gamma_2\}$ *and* $\{\Gamma_1, \Gamma_3\}$ *be info-based complementary pairs, and let* $(M, N) \in \mathsf{LDB}(\mathbf{D}) \times \mathsf{LDB}(\mathbf{V}_1)$ *be an update request from* Γ_1 *to* $\mathbf{D}$ *with the property that both* $\mathsf{CCTransl}\langle (M, N), \Gamma_1; \Gamma_2\rangle$ *and* $\mathsf{CCTransl}\langle (M, N), \Gamma_1; \Gamma_3\rangle$ *exist. Then these two translations are the same; i.e.,* $\mathsf{CCTransl}\langle (M, N), \Gamma_1; \Gamma_2\rangle = \mathsf{CCTransl}\langle (M, N), \Gamma_1; \Gamma_3\rangle$.

Proof. The core idea of the proof is to show that if (M, N) is realizable with constant complement Γ_2 as well as with constant complement Γ_3, then it is realizable with both Γ_2 and Γ_3 constant. This is accomplished by decomposing the realization of (M, N) into a deletion followed by an insertion — something which is in general impossible when working only with complete states but which is possible when atombases are allowed.

Formally, let M_i denote $\mathsf{CCTransl}\langle (M, N), \Gamma_1; \Gamma_i\rangle$ and then define $[M_i'] = M \sqcap M_i$ for $i \in \{2, 3\}$. Think of $[M_i']$ as the state obtained from M by deleting the information in the state of Γ_1 which is not common to both M and M_i. Thus, it represents a deletion of information from M with constant complement Γ_i. Since $(M, [M_i'])$ defines a deletion of information, it can only reduce the information in the state of Γ_{4-i}; it cannot add information to it. More precisely, $\overrightarrow{\gamma_3}([M_2']) \sqsubseteq \overrightarrow{\gamma_3}(M)$ and $\overrightarrow{\gamma_2}([M_3']) \sqsubseteq \overrightarrow{\gamma_2}(M)$. Next, observe that $\overrightarrow{\gamma_i}([M_i']) = \overrightarrow{\gamma_i}(M)$ for $i \in \{2, 3\}$; that is, the update $(M, [M_i'])$ leaves the state of Γ_i constant. Since $[M_i'] \sqsubseteq M$ for $i \in \{2, 3\}$, $[M_2'] \sqcup [M_3'] \sqsubseteq M$ is well defined and in $\mathsf{LAB}(\mathbf{D})$. Letting $[M]$ denote $[M_2'] \sqcup [M_3']$, it follows that $\overrightarrow{\gamma_i}([M']) = \overrightarrow{\gamma_i}(M)$ for $i \in \{2, 3\}$; i.e., $[M']$ keeps the states of both Γ_2 and Γ_3 constant.

The claim is that $((\gamma_1)_{\mathsf{SmEmb}}(N)) \sqcup [M']$ exists and is equal to both $\mathsf{CCTransl}\langle (M, N), \Gamma_1; \Gamma_2\rangle$ and $\mathsf{CCTransl}\langle (M, N), \Gamma_1; \Gamma_3\rangle$. The key is to establish that $\overrightarrow{\gamma_1}([M_i']) \sqcup \overrightarrow{\gamma_i}([M_i']) = [M_i']$ for $i \in \{2, 3\}$. Even though $\{\Gamma_1, \Gamma_i\}$ is a complementary pair, this is not quite immediate, since $[M_i']$ is not necessarily in $\mathsf{LDB}(\mathbf{D})$. However, the elements of $\mathsf{VarsOf}([M_i'])$ occur only in $\overrightarrow{\gamma_1}([M_i'])$ and not in $\overrightarrow{\gamma_i}([M_i'])$, since the latter equals $\overrightarrow{\gamma_i}(M)$. Thus, the problems noted in the example of Definition 47 cannot occur. More formally, $\overrightarrow{\gamma_1}([M_i']) \sqcup \overrightarrow{\gamma_i}([M_i']) = [M_i']$ can be established by replacing $[M_i']$ with any free completion which excludes $\mathsf{ConstOf}([M_i']) \cup \mathsf{ConstOf}(\mathsf{Constr}(\mathbf{D}))$, and noting that the new constants occur only in $\overrightarrow{\gamma_1}([M_i'])$. Furthermore, $\overrightarrow{\gamma_1}([M_2']) = \overrightarrow{\gamma_1}([M_3']) = \overrightarrow{\gamma_1}([M'])$, since they all define the same deletion on Γ_1. Then, since $[M_i']$ and $[M']$ agree on both Γ_1 and Γ_i, and given that $\{\Gamma_1, \Gamma_i\}$ is a complementary pair for $i \in \{2, 3\}$, $[M_2'] = [M_3'] = [M']$. Since $(\gamma_1)_{\mathsf{SmEmb}}(N) \sqcup [M_i'] = \mathsf{CCTransl}\langle (M, N), \Gamma_1; \Gamma_i\rangle$, it follows that $(\gamma_1)_{\mathsf{SmEmb}}(N) \sqcup [M'] = \mathsf{CCTransl}\langle (M, N), \Gamma_1; \Gamma_i\rangle$ as well; in particular $\mathsf{CCTransl}\langle (M, N), \Gamma_1; \Gamma_2\rangle = \mathsf{CCTransl}\langle (M, N), \Gamma_1; \Gamma_3\rangle$. □

Remarks 50. The example of Definition 48 provides a simple situation in which the above theorem applies to yield a reflection of the view update which is independent of the choice of complement, but which is not order based. Thus, the above result provides a meaningful extension to [Hegner, 2004, Thm. 4.3].

This problem was also approached from an information perspective in [Hegner, 2008c], in which the core idea was that a constant-complement update minimizes information change. However, it was subsequently shown in [Hegner, 2009, Disc. 4.7] that while the basic idea is sound, more elaborate constructions are required to avoid so-called collateral changes. The approach given here is substantially less complex than that of those two papers.

A natural question to ask is whether there are interesting examples of complementary pairs which are not info-based pairs, and so the theory of this section does not apply. The answer appears to be a qualified yes; the examples of [Hegner, 2004, Example 4.8], which involve afunctional dependencies [Paredaens et al., 1989, Def, 5.4] are not of type $\exists\wedge+$, although further work is necessary to determine whether they can be incorporated into an extension of this theory.

In order to support constant-complement view update in a systematic fashion, it is necessary to require a stronger form of complement, known as a *meet complement*. Formally, Γ_1 and Γ_2 are meet complements if their congruences commute ($\mathsf{Congr}(\Gamma) = \{(M, M') \in \mathsf{LDB}(\mathbf{D}) \times \mathsf{LDB}(\mathbf{D}) \mid \gamma_{\mathsf{LDB}}(M) = \gamma_{\mathsf{LDB}}(M')\}$). This condition is equivalent to the existence of a third view, called the *meet* of Γ_1 and Γ_2, which defines their intersection. A pair $(N_1, N_2) \in \mathsf{LDB}(\mathbf{V}_1) \times \mathsf{LDB}(\mathbf{V}_2)$ is the image of some $M \in \mathsf{LDB}(\mathbf{D})$ under $(\gamma_1)_{\mathsf{LDB}} \times (\gamma_2)_{\mathsf{LDB}}$ iff N_1 and N_2 agree on the meet. This condition ensures that a constant-complement update to Γ_1 is independent of the state of Γ_2. See [Hegner, 2004] for details. It is important to extend this work to meet complements because it is within that context that the update results of this section may be applied in a systematic manner to databases containing variables — it is precisely the meet which must be free of variables.

Theorem 49 provides conditions under which a constant-complement translation of a view update is independent of the choice of complement. However, it leaves open the question of whether a single complement can support all view updates which are possible via constant complement. The answer is negative in the general case, with the conditions under which it is possible a property of the underlying schema. This question is investigated further in [Hegner, 2010].

7 Conclusions and Further Directions

A framework which retains the essential flavor of the state-based approach to database modelling, while at the same time allowing for enough representation of partial information to recapture the state of a view within the main schema has been presented. The context includes virtually all database dependencies, and all views defined by existential conjunctive queries. The utility of this approach has been demonstrated by the solution of an important uniqueness problem for constant-complement update. There are nevertheless many directions in which these basic ideas should be extended.

Generalized selection and attribute types The framework of this paper supports views defined by simple selections such as $\sigma_{A=\mathrm{a}}(R)$, but it does not include range selections such as $\sigma_{A\leq 10}(R)$ or disjunctions such as $\sigma_{A=\mathrm{a}_1\vee A=\mathrm{a}_2}(R)$. The atombase approach is simply not powerful enough to represent ranges or disjunction, and attempts to "tag" such constants with ranges introduce difficulties, particularly with respect to the essential tool of free completion. For the range query, one possible approach would be to introduce $\leq$ as a relation within the schema. This relation will be infinite, but since it is fixed, that problem can be overcome. For a disjunctive query, an approach would be to allow unary "type" relations, and then put the desired values into that relation. Thus, for the above query, the user would first fill the type relation τ with $\{\tau(\mathrm{a}_1), \tau(\mathrm{a}_2)\}$, and then use a view definition of the form $R(\ldots x_A \ldots) \wedge \tau(x_A)$. There are of course many details to be worked out.

Extension to other first-order data models In recent years, data models which allow for nested relations, yet which are representable within first-order logic, such as the HERM model [Thalheim, 2000], have found increasing use in database modelling. It is therefore important to explore the possibility of extending this framework to such models.

The lattice of views The lattice operations for database states introduced in Section 5, in combination with the internalization of views which is the focus of this paper may be combined naturally to define a lattice of views. Such a lattice would have many uses, including the characterization of optimal decompositions and the appropriate architecture of database components [Thalheim, 2005], [Hegner, 2008b], and thus is worthy of further investigation.

Acknowledgments

Anonymous referees provided suggestions which simplified and improved the presentation substantially.

References

[Abiteboul et al., 1995] Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases; Addison-Wesley (1995).

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

[Chandra and Merlin, 1977] Chandra, A. K., Merlin, P. M.: "Optimal implementation of conjunctive queries in relational data bases"; "Proceedings of the Ninth Annual ACM Symposium on Theory of Computing, 2-4 May 1977, Boulder, Colorado", (1977); 77–90.

[Chang and Lee, 1973] Chang, C.-L., Lee, R. C.-T.: Symbolic Logic and Mechanical Theorem Proving; Academic Press (1973).

[Fagin, 1982] Fagin, R.: "Horn clauses and database dependencies"; J. Assoc. Comp. Mach.; 29, 4 (1982), 952–985.

[Fagin et al., 2005] Fagin, R., Kolaitis, P. G., Miller, R. J., Popa, L.: "Data exchange: Semantics and query answering"; Theoret. Comput. Sci.; 336 (2005), 89–124.

[Genesereth and Nilsson, 1987] Genesereth, M. R., Nilsson, N. J.: Logical Foundations of Artificial Intelligence; Morgan-Kaufmann (1987).

[Grätzer, 1978] Grätzer, G.: General Lattice Theory; Academic Press (1978).

[Hegner, 2004] Hegner, S. J.: "An order-based theory of updates for closed database views"; Ann. Math. Art. Intell.; 40 (2004), 63–125.

[Hegner, 2006] Hegner, S. J.: "The complexity of embedded axiomatization for a class of closed database views"; Ann. Math. Art. Intell.; 46 (2006), 38–97.

[Hegner, 2008a] Hegner, S. J.: Information-Based Distance Measures and the Canonical Reflection of View Updates; Technical Report 0805; Institut für Informatik, Christian-Albrechts-Universität zu Kiel (2008); an updated and corrected version is available on the Web site of the author.

[Hegner, 2008b] Hegner, S. J.: "A model of database components and their interconnection based upon communicating views"; Jakkola, H., Kiyoki, Y., Tokuda, T., editors, "Information Modelling and Knowledge Systems XIX", Frontiers in Artificial Intelligence and Applications; IOS Press (2008); 79–100.

[Hegner, 2008c] Hegner, S. J.: "Semantic bijectivity and the uniqueness of constant-complement updates in the relatiional context"; Schewe, K.-D., Thalheim, B., editors, "International Workshop on Semantics in Data and Knowledge Bases, SDKB 2008, Nantes, France, March 29, 2008, Proceedings", Springer-Verlag (2008); Lecture Notes in Computer Science, volume 4925; 172–191.

[Hegner, 2009] Hegner, S. J.: "Optimal reflection of bidirectional view updates using information-based distance measures"; Bertossi, L., Christiansen, H., editors, "Proceedings of the International Workshop on Logic in Databases (LID 2009), Roskilde University, Denmark, October 29, 2009", (2009); 55–70.

[Hegner, 2010] Hegner, S. J.: "Algebraic characterization of optimal complements of database views"; Schewe, K.-D., Thalheim, B., editors, "Fourth International Workshop on Semantics in Data and Knowledge Bases, SDKB 2010, Bordeaux, France, July 5, 2010, Proceedings", Springer-Verlag (2010); Lecture Notes in Computer Science, volume TBA.

[Horn, 1951] Horn, A.: "On sentences which are true of direct unions of algebras"; J. Symbolic Logic; 16 (1951), 14–21.

[Hull, 1984] Hull, R.: "Finitely specifiable implicational dependency families"; J. Assoc. Comp. Mach.; 31, 2 (1984), 210–226.

[Jacobs et al., 1982] Jacobs, B. E., Aronson, A. R., Klug, A. C.: "On interpretations of relational languages and solutions to the implied constraint problem"; ACM Trans. Database Systems; 7, 2 (1982), 291–315.

[Maher and Srivastava, 1996] Maher, M. J., Srivastava, D.: "Chasing constrained tuple-generating dependencies"; "Proceedings of the Fifteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 3-5, 1996, Montreal, Canada", ACM Press (1996); 128–138.

[Maier et al., 1979] Maier, D., Mendelzon, A. O., Sagiv, Y.: "Testing implications of data dependencies"; ACM Trans. Database Systems; 4, 4 (1979), 455–469.

[Meier et al., 2009] Meier, M., Schmidt, M., Lausen, G.: "On chase termination beyond stratification"; Proc. VLDB Endow.; 2, 1 (2009), 970–981; extended technical report: CoRR, abs/0906.4228.

[Minker, 1987] Minker, J., editor: Foundations of Deductive Databases and Logic Programming; Morgan Kaufmann (1987).
[Monk, 1976] Monk, J. D.: Mathematical Logic; Springer-Verlag (1976).
[Paredaens et al., 1989] Paredaens, J., De Bra, P., Gyssens, M., Van Gucht, D.: The Structure of the Relational Database Model; Springer-Verlag (1989).
[Thalheim, 2000] Thalheim, B.: Entity-Relationship Modeling; Springer-Verlag (2000).
[Thalheim, 2005] Thalheim, B.: "Component development and construction for database design"; Data Knowl. Eng.; 54, 1 (2005), 77–95.

Brief Index of Notation and Terminology

The indices are paragraph numbers, not page numbers. Thus, $\perp$ is found in Definition 4, which is not necessarily on page 4.