ELSEVIER
Available online at www.sciencedirect.com
SCIENCE DIRECT®
Electronic Notes in Theoretical Computer Science 143 (2006) 73–85
Electronic Notes in Theoretical Computer Science
www.elsevier.com/locate/entcs

An Equivalence between Dependencies in Nested Databases and a Fragment of Propositional Logic

Sven Hartmann[1] and Sebastian Link[2]

Information Science Research Centre, Department of Information Systems
Massey University, Palmerston North, New Zealand

Abstract

We extend the result on the equivalence between functional and multivalued dependencies in relational databases and a fragment of propositional logic. It is shown that this equivalence still holds for functional and multivalued dependencies in databases that support complex values via nesting of records and lists.
The equivalence has several implications. Firstly, it extends a well-known result from relational databases to databases which are not in first normal form. Secondly, it characterises the implication of functional and multivalued dependencies in complex-value databases in purely logical terms. The database designer can take advantage of this equivalence to reduce database design problems to simpler problems in propositional logic. Furthermore, relational database design tools can be reused to solve problems for complex-value databases.

Keywords: Logic in Databases, Nested Databases, Lists, Functional Dependency, Multivalued Dependency, Brouwerian Algebra, Propositional Logic

1 Introduction

Functional dependencies (FDs,[6]) and multivalued dependencies (MVDs,[12,7]) are fundamental and widely studied concepts in relational database theory. While the notion of an FD is intuitively simple, MVDs are more general than FDs and characterise precisely those database instances that can be decomposed without loss of information. According

[1] Email:s.hartmann@massey.ac.nz
[2] Email:s.link@massey.ac.nz

doi:10.1016/j.entcs.2005.04.039

to [8] about three quarters of all uni-relational dependencies (dependencies over a single relation schema) defined in practice are FDs and MVDs. It is well-known that the implication of FDs and MVDs is equivalent to the logical implication of formulae in a certain fragment of propositional logic [21]. It is therefore possible to take advantage of research in the area of propositional logic by converting familiar results into results about relational dependencies. The equivalence has resulted in several applications [20,19,9]. In particular, the equivalence reduces to the fragment of Horn clauses when FDs are studied by themselves [11].

Many researchers have remarked that classical database design problems need to be revisited in new data formats [22,23]. Biskup [5] has listed two particular challenges for database design theory: finding a unifying framework and extending achievements to deal with advanced database features such as complex type constructors. One possibly unifying framework can result from the classification of data models according to the type constructors that are supported by the model. The relational data model can be captured by a single application of the record constructor, arbitrary nesting of record and set constructor covers aggregation and grouping which are fundamental to many semantic data models as well as the nested relational data model. The Entity-Relationship Model and its extensions require record, set and (disjoint) union constructor. A minimal set of type constructors supported by any object-oriented data model includes record, list, set and multiset (bag) constructor. Genomic sequence data models call for support of records, lists and sets. Finally, XML requires at least record (concatenation), list (Kleene Closure), union (optionality), and reference constructor.

The major goal of this paper is to generalise the Equivalence theorem from [21] to nested databases that support record and list constructor. Further need for lists arises from applications that store ordered relations, time-series data, meteorological and astronomical data streams, runs of experimental data, multidimensional arrays, textual information, voices, sound, images, video, etc. Our studies will be based on an abstract data model that defines a database schema as a nested attribute that results from flat attributes by any finite number of recursive applications of record and list constructor. It is our intention not to focus on the specifics of any particular data model in order to place emphasis on the type constructors themselves. FDs and MVDs have previously been defined in terms of subschemata of the underlying database schema. This approach leads to Brouwerian algebras [18] and provides therefore a mathematically well-founded framework that is sufficiently flexible and powerful to study design problems for different classes of dependencies with respect to different combinations of type constructors.

2 Nested Attributes

Complex-value data models have been proposed to overcome severe limitations of the RDM when designing many practical database applications [1]. In the following, we study the underlying algebra of nested database schemata that support record and list constructor. This can be extended to include constructors for sets and multisets as well [16]. We start with the definition of flat attributes and values for them.

A *universe* is a finite set $\mathcal{U}$ together with domains (i.e. sets of values) $dom(A)$ for all $A \in \mathcal{U}$. The elements of $\mathcal{U}$ are called *flat attributes*. For the relational data model a universe was sufficient. That is, a relation schema is defined as a finite subset $R \subseteq \mathcal{U}$. For data models supporting type constructors, however, nested attributes may be utilised. In the following definition we use a set $\mathcal{L}$ of labels, and assume that the symbol λ is neither a flat attribute nor a label, i.e., $\lambda \notin \mathcal{U} \cup \mathcal{L}$. Moreover, flat attributes are not labels and vice versa, i.e., $\mathcal{U} \cap \mathcal{L} = \emptyset$.

Let $\mathcal{U}$ be a universe and $\mathcal{L}$ a set of labels. The set $\mathcal{N}\mathrm{A}(\mathcal{U}, \mathcal{L})$ of *nested attributes over* $\mathcal{U}$ *and* $\mathcal{L}$ is the smallest set satisfying the following conditions: $\lambda \in \mathcal{N}\mathrm{A}(\mathcal{U}, \mathcal{L})$, $\mathcal{U} \subseteq \mathcal{N}\mathrm{A}(\mathcal{U}, \mathcal{L})$, for $L \in \mathcal{L}$ and $N_1, \ldots, N_k \in \mathcal{N}\mathrm{A}(\mathcal{U}, \mathcal{L})$ with $k \geq 1$ we have $L(N_1, \ldots, N_k) \in \mathcal{N}\mathrm{A}(\mathcal{U}, \mathcal{L})$, and for $L \in \mathcal{L}$ and $N \in \mathcal{N}\mathrm{A}(\mathcal{U}, \mathcal{L})$ we have $L[N] \in \mathcal{N}\mathrm{A}(\mathcal{U}, \mathcal{L})$. We call λ *null attribute*, $L(N_1, \ldots, N_k)$ *record-valued attribute*, and $L[N]$ *list-valued attribute*. From now on we will assume that a universe $\mathcal{U}$ and a set $\mathcal{L}$ of labels are fixed, and drop the index simply writing $\mathcal{N}\mathrm{A}$ instead of $\mathcal{N}\mathrm{A}(\mathcal{U}, \mathcal{L})$.

The relation schema $R = \{A_1, \ldots, A_k\}$ with flat attributes A_i may now be considered as the record-valued attribute $R(A_1, \ldots, A_k)$ where R is used as a label. For instance, the relation schema DVD={Title, Actor, Feature} becomes the record-valued nested attribute DVD(Title,Actor,Feature) in which the name DVD of the relation schema is used as a label.

We can now extend the mapping *dom* from flat attributes to nested attributes, i.e., we define a set $dom(N)$ of values for every nested attribute $N \in \mathcal{N}\mathrm{A}$. The empty list is denoted by [ ]. For a nested attribute $N \in \mathcal{N}\mathrm{A}$ we define the *domain* $dom(N)$ as follows: $dom(\lambda) = \{ok\}$, $dom(L(N_1, \ldots, N_k)) = \{(v_1, \ldots, v_k) \mid v_i \in dom(N_i) \text{ for } i = 1, \ldots, k\}$, i.e., the set of all k-tuples $(v_1, \ldots, v_k)$ with $v_i \in dom(N_i)$ for all $i = 1, \ldots, k$, and $dom(L[N]) = \{[v_1, \ldots, v_n] \mid v_i \in dom(N) \text{ for } i = 1, \ldots, n \text{ and where } n \in \mathbb{N}\} \cup \{[\,]\}$, i.e., $dom(L[N])$ is the set of all finite lists with elements in $dom(N)$.

The null attriute λ is a distinguished attribute whose domain is the single null value *ok* which indicates that some information exists but has currently been left out. For example, we may refer to the attribute set {Actor} of the

relation schema DVD by using the nested attribute DVD(λ,Actor,λ). That is, the information on the Title and the Feature is omitted.

The previous example demonstrates how the replacement of flat attribute names by the null attribute λ within a nested attribute decreases the amount of information that is modelled by the corresponding attributes. This fact allows to introduce an order $\leq$ between nested attributes which generalises the inclusion order on sets of flat attributes from the relational model.

The *subattribute relation* $\leq$ on the set of nested attributes $\mathcal{N}\mathrm{A}$ over $\mathcal{U}$ and $\mathcal{L}$ is defined by the following rules, and the following rules only: (i) $N \leq N$ for all nested attributes $N \in \mathcal{N}\mathrm{A}$, (ii) $\lambda \leq A$ for all flat attributes $A \in \mathcal{U}$, (iii) $\lambda \leq N$ for all list-valued attributes $N \in \mathcal{N}\mathrm{A}$, (iv) $L(N_1, \ldots, N_k) \leq L(M_1, \ldots, M_k)$ whenever $N_i \leq M_i$ for all $i = 1, \ldots, k$, and (v) $L[N] \leq L[M]$ whenever $N \leq M$. For $N, M \in \mathcal{N}\mathrm{A}$ we say that M is a *subattribute* of N if and only if $M \leq N$ holds. We write $M \not\leq N$ if and only if M is not a subattribute of N, and write $M < N$ if and only if $M \leq N$ but $M \neq N$.

According to the second and third rule the null attribute is not defined to be a subattribute of an arbitrary record-valued attribute. The record-valued attribute DVD(λ,λ,λ) already represents a minimal amount of information, i.e., all elements from the domain of DVD(Title,Actor,Feature) have the same projection on DVD(λ,λ,λ). Therefore, λ does not need to be defined as a subattribute of it.

The subattribute relation $\leq$ on nested attributes is reflexive, antisymmetric and transitive, i.e., a partial order on nested attributes. Informally, $M \leq N$ for $N, M \in \mathcal{N}\mathrm{A}$ if and only if M comprises at most as much information as N does. The informal description of the subattribute relation is formally documented by the existence of a projection function $\pi_M^N : dom(N) \to dom(M)$ in case $M \leq N$ holds.

Let $N, M \in \mathcal{N}\mathrm{A}$ with $M \leq N$. The *projection function* $\pi_M^N : dom(N) \to dom(M)$ is defined as follows: if $N = M$, then $\pi_M^N = id_{dom(N)}$ is the identity on $dom(N)$, if $M = \lambda$, then $\pi_\lambda^N : dom(N) \to \{ok\}$ is the constant function that maps every $v \in dom(N)$ to ok, if $N = L(N_1, \ldots, N_k)$ and $M = L(M_1, \ldots, M_k)$, then $\pi_M^N = \pi_{M_1}^{N_1} \times \cdots \times \pi_{M_k}^{N_k}$ which maps every tuple $(v_1, \ldots, v_k) \in dom(N)$ to $(\pi_{M_1}^{N_1}(v_1), \ldots, \pi_{M_k}^{N_k}(v_k)) \in dom(M)$, and if $N = L[N']$ and $M = L[M']$, then $\pi_M^N : dom(N) \to dom(M)$ maps every list $[v_1, \ldots, v_n] \in dom(N)$ to the list $[\pi_{M'}^{N'}(v_1), \ldots, \pi_{M'}^{N'}(v_n)] \in dom(M)$.

It follows, in particular, that the empty list [ ] is always mapped to ifself, except when projected on the null attribute λ in which it is mapped to ok. Let $\circ$ denote the composition of functions. Then we have the following fundamental fact. Let $N \in \mathcal{N}\mathrm{A}$. For all $X, Y \in Sub(N)$ with $Y \leq X$ we have $\pi_Y^N = \pi_Y^X \circ \pi_X^N$. This shows in particular that two elements from the domain

of N which have the same projection on X also have the same projection on any subattribute Y of X.

Dependency theory in the relational data model is based on the powerset $\mathcal{P}(R)$ for a relation schema R. In fact, $\mathcal{P}(R)$ is a powerset algebra with set inclusion $\subseteq$, set union $\cup$, set intersection $\cap$ and set difference $-$. Having fixed some nested attribute N one may consider the finite set $Sub(N)$ of all its *subattributes*, namely $Sub(N) = \{M \mid M \leq N\}$. The restriction of $\leq$ to $Sub(N)$ is a partial order on $Sub(N)$.

We investigate the algebraic structure of $(Sub(N), \leq)$. A *Brouwerian algebra* [18] is a lattice $(L, \sqsubseteq, \sqcup, \sqcap, \dot{-}, 1)$ with top element 1 and a binary operation $\dot{-}$ which satisfies $a \dot{-} b \sqsubseteq c$ iff $a \sqsubseteq b \sqcup c$ for all $c \in L$. In this case, the operation $\dot{-}$ is called the *pseudo-difference*. The *Brouwerian complement* $\neg a$ of $a \in L$ is then defined by $\neg a = 1 \dot{-} a$.

The following theorem generalises the fact that $(\mathcal{P}(R), \subseteq, \cup, \cap, -, \emptyset, R)$ is a Boolean algebra for a relation schema R in the RDM. Note that the operations of join $\sqcup_N$ and meet $\sqcap_N$ are determined by the previous definition of the subattribute relation $\leq$. The operation of pseudo-difference $\dot{-}_N$ is determined by the definition of $\leq$ and $\sqcup_N$.

Theorem 2.1 *$(Sub(N), \leq, \sqcup_N, \sqcap_N, \dot{-}_N, N)$ forms a Brouwerian algebra for every $N \in \mathcal{NA}$.* □

Given some nested attribute $N \in \mathcal{NA}$ and $Y, Z \in Sub(N)$, we use $Y_N^{\mathcal{C}} = N \dot{-} Y$ to denote the *Brouwerian complement* of Y in $Sub(N)$. The algebra $(Sub(N), \leq, \sqcup, \sqcap, (\cdot)^{\mathcal{C}}, \lambda_N, N)$ is in general not Boolean. If the context allows we will omit the index N from the operations $\sqcup_N, \sqcap_N, \dot{-}_N$ and from λ_N as well as $Y_N^{\mathcal{C}}$. Fundamental is the fact that the projection of an element $t \in dom(N)$ on two subattributes $X, Y \in Sub(N)$ always determines the projection on the join $X \sqcup Y$.

Lemma 2.2 *Let $X, Y \in Sub(N)$ and $t_1, t_2 \in dom(N)$. If $\pi_X^N(t_1) = \pi_X^N(t_2)$ and $\pi_Y^N(t_1) = \pi_Y^N(t_2)$, then $\pi_{X \sqcup Y}^N(t_1) = \pi_{X \sqcup Y}^N(t_2)$.* □

Lemma 2.2 suggests to focus on the set $SubB(N)$ of join-irreducible elements of $(Sub(N), \leq, \sqcup, \sqcap, \lambda_N)$. Recall that an element a of a lattice with bottom element 0 is called *join-irreducible* if and only if $a \neq 0$ and if $a = b \sqcup c$ holds for any elements b and c, then $a = b$ or $a = c$.

3 Functional and Multivalued Dependencies

We repeat fundamental definitions and results from [16,17].

Definition 3.1 *Let $N \in \mathcal{NA}$ be a nested attribute. A functional dependency*

on N is an expression of the form $X \to Y$ where $X, Y \in Sub(N)$. A set $r \subseteq dom(N)$ satisfies the functional dependency $X \to Y$ on N, denoted by $\models_r X \to Y$, if and only if for all $t_1, t_2 \in r$ the following holds: if $\pi_X^N(t_1) = \pi_X^N(t_2)$, then $\pi_Y^N(t_1) = \pi_Y^N(t_2)$. □

We consider the following examples to become more acquainted with FDs.

Example 3.2 *Suppose we use Shop(Person,Store[Purchase(Article,Price)]) to store sequences of purchases by a person that buys articles for a certain price. A snapshot r of such a database may look as follows:*

{ *(Nicki, [(Bag, 59.85), (Shoes, 98.99)]),*
(Nicki, [(Bag, 68.95), (Shoes, 119.95)]),
(Nicole, [ ]),
(Paris, [(Shoes, 169.97), (Skirt, 99.25), (Shoes, 395.85)]),
(Paris, [(Shoes, 123.45), (Skirt, 109.75), (Shoes, 212.15)]) }.

In r the same person always buys the same articles in the same order, i.e. $\models_r$ Shop(Person,λ) → Shop(λ,Store[Purchase(Article,λ)]). □

As it was the case for FDs the algebraic framework allows a natural extension of the definition of multivalued dependencies from the RDM.

Definition 3.3 *Let $N \in \mathcal{N}A$ be a nested attribute. A multivalued dependency on N is an expression of the form $X \twoheadrightarrow Y$ where $X, Y \in Sub(N)$. A set $r \subseteq dom(N)$ satisfies the multivalued dependency $X \twoheadrightarrow Y$ on N if and only if for all values $t_1, t_2 \in r$ with $\pi_X^N(t_1) = \pi_X^N(t_2)$ there is a value $t \in r$ with $\pi_{X \sqcup Y}^N(t) = \pi_{X \sqcup Y}^N(t_1)$ and $\pi_{X \sqcup Y^C}^N(t) = \pi_{X \sqcup Y^C}^N(t_2)$.* □

Intuitively, an instance r exhibits the MVD $X \twoheadrightarrow Y$ whenever the value on X determines the set of values on Y independently from the set of values on Y^C. We will illustrate the concept of an MVD by the following example.

Example 3.4 *Consider Shop(Person,Store[Purchase(Article,Price)]) that was used as a nested attribute in Example 3.2. Suppose r is now*

{ *(Nicki, [(Bag, 59.85), (Shoes, 98.99)]),*
(Nicki, [(Bag, 68.95), (Shoes, 119.95)]),
(Paris, [(Shoes, 169.97), (Skirt, 99.25), (Shoes, 395.85)]),
(Paris, [(Top, 169.97), (Hat, 99.25), (Pants, 395.85)]),
(Paris, [(Shoes, 123.45), (Skirt, 109.75), (Shoes, 212.15)]),
(Paris, [(Top, 123.45), (Hat, 109.75), (Pants, 395.85)]),
(Nicole, [ ]) } .

Obviously, the FD Shop(Person,λ) → Shop(λ,Store[Purchase(Article,λ)])

is not satisfied by r, and neither is the FD Shop(Person,λ) → Shop(λ,Store[Purchase(λ,Price)]). However, $\models_r$ *Shop(Person,λ)* $\twoheadrightarrow$ *Shop(λ,Store[Purchase(Article,λ)]). That is, the sequence of articles a person purchases is independent from the sequence of prices in this snapshot. It appears that* $\models_r$ *Shop(Person,λ) → Shop(λ,Store[Purchase(λ,λ)]) holds as well. This means informally that in this snapshot each person makes a fixed number of purchases.* □

MVDs from the RDM appear now as special cases. For example, the MVD Title ↠ Actor on the relation schema DVD becomes the DVD(Title,λ,λ) ↠ DVD(λ,Actor,λ) on the nested attribute DVD(Title,Actor,Feature).

Fagin proves in [12] that MVDs "provide a necessary and sufficient condition for a relation to be decomposable into two of its projections without loss of information (in the sense that the original relation is guaranteed to be the join of the two projections)." In order to generalise this desirable property, we define the generalised natural join within our framework. Let $N \in \mathcal{NA}$ and $X, Y \in Sub(N)$. Let $r_1 \subseteq dom(X)$ and $r_2 \subseteq dom(Y)$. Then $r_1 \bowtie r_2 = \{t \in dom(X \sqcup Y) \mid \exists t_1 \in r_1, t_2 \in r_2.\pi_X^{X \sqcup Y}(t) = t_1 \text{ and } \pi_Y^{X \sqcup Y}(t) = t_2\}$ is called the *generalised natural join* $r_1 \bowtie r_2$ *of* r_1 and r_2. The *projection* $\pi_X(r)$ of $r \subseteq dom(N)$ on $X \in Sub(N)$ is defined as $\{\pi_X^N(t) \mid t \in r\}$.

Theorem 3.5 *Let* $N \in \mathcal{NA}$, $r \subseteq dom(N)$ *and* $X \twoheadrightarrow Y$ *a multivalued dependency on N. Then* $X \twoheadrightarrow Y$ *is satisfied by r if and only if* $r = \pi_{X \sqcup Y}(r) \bowtie \pi_{X \sqcup Y^C}(r)$. □

Let $\mathcal{C}$ denote a class of dependencies, Σ be a set of dependencies from $\mathcal{C}$, and σ a single dependency from $\mathcal{C}$, all defined on some nested attribute N. We say that Σ (finitely) implies σ, denoted by $\Sigma \models \sigma$ ($\Sigma \models_{\text{fin}} \sigma$) if and only if all (finite) $r \subseteq dom(N)$ that satisfy all dependencies in Σ also satisfy σ. Furthermore, Σ implies σ in the world of two-element instances if and only if all $r = \{t_1, t_2\} \subseteq dom(N)$ that satisfy all dependencies in Σ also satisfy σ. We deal with the class $\mathcal{C}$ of FDs and MVDs for which finite and unrestricted implication coincide [17]. As it will turn out in this paper, (finite) implication for $\mathcal{C}$ even coincides with implication in the world of two-element instances.

4 The Equivalence Result

We assume familiarity with basic notions from classical Boolean propositional logic [10]. We will be particularly interested in propositional formulae of the form $(U_1 \wedge \cdots \wedge U_n) \Rightarrow (V_1 \wedge \cdots \wedge V_m)$ or $(U_1 \wedge \cdots \wedge U_n) \Rightarrow ((V_1 \wedge \cdots \wedge V_m) \vee (W_1 \wedge \cdots \wedge W_k))$ where U_i, V_j, W_l denote propositional variables. We assume that the conjunction of 0 propositional variables is *true*, and that

conjunction $\wedge$ and disjunction $\vee$ bind stronger than implication $\Rightarrow$. The formulae above become $U_1 \wedge \cdots \wedge U_n \Rightarrow V_1 \wedge \cdots \wedge V_m$ and $U_1 \wedge \cdots \wedge U_n \Rightarrow (V_1 \wedge \cdots \wedge V_m) \vee (W_1 \wedge \cdots \wedge W_k)$, respectively. The satisfaction of a propositional formula ϕ by a truth assignment θ is denoted by $\models_\theta \phi$.

4.1 An Example

Consider the nested attribute Shop(Person,Store[Purchase(Article,Price)]) on which the MVD σ: Shop(Person,λ) $\twoheadrightarrow$ Shop(λ,Store[Purchase(Article,λ)]) has been specified by the designer of the database. One may ask whether one of the FDs σ_1: Shop(Person,λ) $\rightarrow$ Shop(λ,Store[Purchase(Article,λ)]) or σ_2: Shop(Person,λ) $\rightarrow$ Shop(λ,Store[Purchase(λ,λ)]) are implied by σ. Consider for instance the two elements

(Paris, [(Shoes, 169.97), (Skirt, 99.25), (Shoes, 395.85)]), and
(Paris, [(Top, 169.97), (Hat, 99.25), (Pants, 395.85)])

in which both elements have the same projection on Shop(Person,λ) and Shop(λ,Store[Purchase(λ,Price)]), but different projections on Shop(λ,Store[Purchase(Article,λ)]). Therefore, σ does not imply σ_1. In order to see that σ does imply σ_2 consider any instance that satisfies σ, and any two tuples t_1, t_2 in this instance that have the same projection on Shop(Person,λ). Since σ is satisfied, there must be another tuple in this instance that has the same projection as the first tuple on Shop(Person,Store[Purchase(Article,λ)]) and the same projection as the second tuple on Shop(Person,Store[Purchase(λ,Price)]). Since Shop(λ,Store[Purchase(λ,λ)]) is a subattribute of both Shop(Person,Store[Purchase(Article,λ)]) and Shop(Person,Store[Purchase(λ,Price)]) the two tuples t_1, t_2 must have the same projection on it, i.e., σ_2 is also satisfied.

Consider the implication problem from a logical point of view. Therefore, we assign propositional variables to the join-irreducible elements of (Shop(Person,Store[Purchase(Article,Price)]),$\leq$) as follows:

- Shop(Person,λ) is assigned V_1,
- Shop(λ,Store[Purchase(Article,λ)]) is assigned V_2,
- Shop(λ,Store[Purchase(λ,Price)]) is assigned V_3, and
- Shop(λ,Store[Purchase(λ,λ)]) is assigned V_4.

According to the subattribute order $\leq$ between the join-irreducibles, the truth assignments to $V_1,\ldots,V_4$ cannot be independet from one another. In this example, the structure of N is encoded by the formulae $V_2 \Rightarrow V_4$ and $V_3 \Rightarrow V_4$. The MVD and FDs above result in the following propositional formulae $V_1 \Rightarrow V_2 \vee V_3$ (σ), $V_1 \Rightarrow V_2$ (σ_1) and $V_1 \Rightarrow V_4$ (σ_2). This correspondence will be

defined later on in detail. The truth assignment θ with $\theta(V_i) = \mathit{true}$ if and only if $i \in \{1, 3, 4\}$ satisfies $V_1 \Rightarrow V_2 \vee V_3$, but violates $V_1 \Rightarrow V_2$. That is, $V_1 \Rightarrow V_2$ is not logically implied by $V_1 \Rightarrow V_2 \vee V_3$. Note that the two elements in the instance above coincide on projections to precisely those join-irreducibles X whose corresponding propositional variable is interpreted as *true* by θ. It is relatively simple to see that $V_1 \Rightarrow V_4$ is logically implied by $V_1 \Rightarrow V_2 \vee V_3$. Consider any truth assignment that assigns *true* to V_1. Then V_2 or V_3 must be interpreted as true by θ. According to the formulae that encode N it follows that V_4 is interpreted as *true* by θ.

4.2 The Result

For $X \in Sub(N)$ let $\vartheta(X) = \{W \in SubB(N) \mid W \leq X$ and for all $W' \in SubB(N)$ with $W \leq W'$ and $W' \leq X$ follows $W = W'\}$.

Let $\varphi : SubB(N) \to \mathcal{V}$ denote a bijection between the join-irreducible elements of $(Sub(N), \leq)$ and the set $\mathcal{V}$ of propositional variables. Consider the FD σ: $X \to Y$ on N where $\vartheta(X) = \{X_1, \ldots, X_n\}$ and $\vartheta(Y) = \{Y_1, \ldots, Y_k\}$. Define $\Phi(\sigma)$ to be the propositional formula

$$\varphi(X_1) \wedge \cdots \wedge \varphi(X_n) \Rightarrow \varphi(Y_1) \wedge \cdots \wedge \varphi(Y_k) \quad .$$

Consider the MVD $\sigma = X \twoheadrightarrow Y$ on N where $\vartheta(X) = \{X_1, \ldots, X_n\}$, $\vartheta(Y) = \{Y_1, \ldots, Y_k\}$ and $\vartheta(Y_N^{\mathcal{C}} \dot{-} X) = \{Z_1, \ldots, Z_m\}$. Define $\Phi(\sigma)$ to be the propositional formula

$$\varphi(X_1) \wedge \cdots \wedge \varphi(X_n) \Rightarrow (\varphi(Y_1) \wedge \cdots \wedge \varphi(Y_k)) \vee (\varphi(Z_1) \wedge \cdots \wedge \varphi(Z_m)) \quad .$$

The MVD σ: Shop(Person,λ) $\twoheadrightarrow$ Shop(λ,Store[Purchase(Article,λ)]), for example, results in $V_1 \Rightarrow V_2 \vee V_3$. If Σ is a set of FDs and MVDs defined on N, let $\Pi = \{\Phi(\sigma) \mid \sigma \in \Sigma\}$ denote the corresponding set of propositional formulae over $\mathcal{V}$. Furthermore, the set

$$\Pi_N = \{\varphi(U) \Rightarrow \varphi(V) \mid U, V \in SubB(N), U \text{ covers}^{\,3}\, V\}$$

denotes those formulae which encode the structure of N. The main result of this paper is the following extension of the Equivalence Theorem from [21].

Theorem 4.1 *[Equivalence Theorem] Let N be a nested attribute, Σ a set of FDs and MVDs and σ a single FD or MVD on N. Let Π_N denote the propositional formulae which encode the structure of N, and Π denote the corresponding set of propositional formulae for Σ. Then*

[3] U covers V iff $U < V$ and for all $W \in SubB(N)$ with $U \leq W \leq V$ we have $U = W$ or $V = W$, this is just the standard definition of a *cover relation* for posets, see [14]

1) Σ *implies* σ*,*

2) Σ *implies* σ *in the world of two-element instances, and*

3) $\Pi \cup \Pi_N$ *logically implies* $\Phi(\sigma)$

are equivalent. □

4.3 An Outline of the Proof

It is immediate that *1)* implies *2)*. The converse implication follows from the following lemma which is interesting in its own right as a model-theoretical result. The proof of this lemma can be found in the full paper.

Lemma 4.2 *Assume* $r \subseteq dom(N)$ *is some finite instance over* N*,* Σ *a set of FDs and MVDs on* N*, and* σ *a single FD or MVD on* N*. Suppose* r *satisfies all FDs and MVDs in* Σ*, but does not satisfy* σ*. Then there is some* $r' = \{t_1, t_2\} \subseteq r$ *such that* r' *satisfies all FDs and MVDs in* Σ*, but does not satisfy* σ*.* □

It remains to demonstrate the equivalence between *2)* and *3)*. First, we show that *2)* implies *3)*. Suppose $\Pi \cup \Pi_N$ does not logically imply $\Phi(\sigma)$. Then there is some truth assignment θ which makes all formulae in $\Pi \cup \Pi_N$ *true*, but $\Phi(\sigma)$ *false*. Since θ satisfies all formulae in Π_N it cannot be the case that $\theta(U) = true$ and $\theta(V) = false$ for any $U, V \in SubB(N)$ with $V < U$. Let $X^+ = \bigsqcup\{W \in SubB(N) \mid \theta(\varphi(W)) = \text{true}\}$. We choose $\{t_1, t_2\} \subseteq dom(N)$ such that for all $W \in SubB(N)$ we have $\pi^N_W(t_1) = \pi^N_W(t_2)$ if and only if $W \leq X^+$. Such an instance $\{t_1, t_2\}$ always exists according to Lemma 4.3.

Lemma 4.3 *Let* $N \in \mathcal{N}A$*. For each* $X \in Sub(N)$ *there are* $t_1, t_2 \in dom(N)$ *such that* $\pi^N_Y(t_1) = \pi^N_Y(t_2)$ *holds for* $Y \in Sub(N)$ *if and only if* $Y \leq X$*.* □

We then have for all $W \in SubB(N)$ that $\theta(\varphi(W)) = true$ iff $W \leq X^+$ iff $\pi^N_W(t_1) = \pi^N_W(t_2)$. Consequently, $\theta = \theta_{\{t_1,t_2\}}$ where $\theta_{\{t_1,t_2\}}$ is defined as in Lemma 4.4. According to Lemma 4.4 we then know that $\{t_1, t_2\}$ satisfies all FDs and MVDs in Σ, but does not satisfy σ. Hence, σ is not implied by Σ in the world of two-element instances.

Lemma 4.4 *Let* σ *be an FD or MVD on the nested attribute* N*, and* $r = \{t_1, t_2\} \subseteq dom(N)$*. Then* $\models_r \sigma$ *if and only if* $\models_{\theta_r} \Phi(\sigma)$ *where*

$$\theta_r(V) = \begin{cases} true\,, & \text{if } \pi^N_{\varphi^{-1}(V)}(t_1) = \pi^N_{\varphi^{-1}(V)}(t_2) \\ false, & \text{else} \end{cases}$$

for all $V \in \varphi(SubB(N))$*.* □

It remains to show that *3)* implies *2)*. Suppose σ is not implied by Σ in the world of two-element instances. Therefore, there is some $r = \{t_1, t_2\} \subseteq dom(N)$ which satisfies all FDs and MVDs in Σ but violates σ. Consider the truth assignment θ_r from Lemma 4.4. It follows that θ_r makes all formulae in $\Pi \cup \Pi_N$ *true*, but $\Phi(\sigma)$ *false*. That is, $\Phi(\sigma)$ is not logically implied by $\Pi \cup \Pi_N$.

5 Reusing Relational Database Design Tools

A direct consequence of the Equivalence theorem from [21] and the Equivalence Theorem 4.1 is that relational database design tools can be applied to solve problems for nested attributes that are generated from flat attributes by recursive applications of record- and list constructor (and even vice versa).

In fact, the join-irreducible elements of $(Sub(N), \leq)$ are interpreted as flat attributes, i.e. the corresponding relation schema is $R_N = SubB(N)$, each FD σ: $X \to Y$ in Σ becomes the relational FD σ': $\vartheta(X) \to \vartheta(Y)$, and each MVD σ: $X \twoheadrightarrow Y$ in Σ becomes the relational MVD σ': $\vartheta(X) \twoheadrightarrow \vartheta(Y)$. Given Σ on the nested attribute N, the corresponding set of relational FDs and MVDs is

$$\Sigma' = \{\sigma' \mid \sigma \in \Sigma\} \cup \{\varphi(U) \to \varphi(V) \mid U, V \in SubB(N), U \text{ covers } V\}.$$

Corollary 5.1 *Let Σ be a set of FDs and MVDs, and σ be a single FD or MVD, all defined on the nested attribute N. Then σ is implied by Σ if and only if σ' is implied by Σ' on R_N.* □

Consider again N = Shop(Person,Store[Purchase(Article,Price)]). The join-irreducibles are mapped to flat attributes as follows:

- Shop(Person,λ) is assigned A,
- Shop(λ,Store[Purchase(Article,λ)]) is assigned B,
- Shop(λ,Store[Purchase(λ,Price)]) is assigned C, and
- Shop(λ,Store[Purchase(λ,λ)]) is assigned D.

The MVD Shop(Person,λ) $\twoheadrightarrow$ Shop(λ,Store[Purchase(Article,λ)]) translates to $A \twoheadrightarrow B$ on $R_N = \{A, B, C, D\}$. Moreover, the structure of N results in the FDs $B \to D$ and $C \to D$. These three dependencies form Σ'. In order to determine whether Shop(Person,λ) $\to$ Shop(λ,Store[Purchase(λ,λ)]) is implied by the MVD above, one can check whether $A \to D$ is implied by Σ'. Well-known techniques from relational databases [3,13] can be applied to compute the dependency basis $DepB(A) = \{A, B, C, D\}$ and closure $A^+ = \{A, D\}$ of A with respect to Σ'. Since $D \in A^+$ it follows indeed that $A \to D$ is implied by Σ'.

6 Related and Future Work

The presence of the list constructor allows to focus on join-irreducible elements of the underlying Brouwerian algebra (Lemma 2.2). In case of other constructors, such as for set or multiset type, Lemma 2.2 fails, and an extension of the Equivalence theorem [11] for FDs alone becomes already sophisticated, see [15]. The papers [11,21] provide alternative proofs for the original equivalence results for FDs and MVDs. These proofs are syntactical in the sense that they take advantage of the fact that the implication of FDs and MVDs can be characterised by finite, sound and complete sets of syntactic inference rules [2,4]. Such sets of inference rules have also been proposed for the setting of the present paper, see the appendix. A generalisation of these syntactical proofs seems desirable. It is future work to further exploit the application of relational database design tools for complex-value database design problems, especially for the proposal of syntactic normal forms that describe semantically well-designed databases such as Fourth normal form for relational databases [12]. We would like to investigate FDs and MVDs in the presence of set, multiset and union constructor.

References

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

[2] Armstrong, W. W., *Dependency structures of database relationships*, Information Processing (1974), pp. 580–583.

[3] Beeri, C., *On the membership problem for functional and multivalued dependencies in relational databases*, Transactions on Database Systems (TODS) **5** (1980), pp. 241–259.

[4] Beeri, C., R. Fagin and J. H. Howard, *A complete axiomatization for functional and multivalued dependencies in database relations*, in: *International Conference on Management of Data (SIGMOD)*, ACM, 1977, pp. 47–61.

[5] Biskup, J., *Achievements of relational database schema design theory revisited*, in: *Semantics in databases*, number 1358 in Lecture Notes in Computer Science (1998), pp. 29–54.

[6] Codd, E. F., *A relational model of data for large shared data banks*, Communications of the ACM **13** (1970), pp. 377–387.

[7] Delobel, C., *Normalisation and hierarchical dependencies in the relational data model*, Transactions on Database Systems (TODS) **3** (1978), pp. 201–222.

[8] Delobel, C. and M. Adiba, "Relational database systems," North Holland, 1985.

[9] Delobel, C. and D. Parker, *Functional and multivalued dependencies in a relational database and the theory of Boolean switching functions*, Technical report, University of Grenoble, Grenoble, France (1978).

[10] Enderton, H., "A mathematical introduction to logic," Academic Press, 1972.

[11] Fagin, R., *Functional dependencies in a relational data base and propositional logic*, IBM Journal of Research and Development **21** (1977), pp. 543–544.

[12] Fagin, R., *Multivalued dependencies and a new normal form for relational databases*, Transactions on Database Systems (TODS) **2** (1977), pp. 262–278.

[13] Galil, Z., *An almost linear-time algorithm for computing a dependency basis in a relational database*, Journal of the ACM **29** (1982), pp. 96–102.

[14] Grätzer, G., "General Lattice Theory," Birkhäuser, 1998.

[15] Hartmann, S. and S. Link, *Horn clauses and functional dependencies in complex-value databases*, see http://infosys.massey.ac.nz/%7Eslink/publ.html.

[16] Hartmann, S., S. Link and K.-D. Schewe, *Axiomatisations of functional and multivalued dependencies in the presence of records, lists, sets and multisets*, accepted for Theoretical Computer Science (TCS).

[17] Hartmann, S., S. Link and K.-D. Schewe, *Functional and multivalued dependencies in nested databases generated by record and list constructor*, accepted for Annals of Mathematics and Artificial Intelligence (AMAI).

[18] McKinsey, J. C. C. and A. Tarski, *On closed elements in closure algebras*, Annals of Mathematics **47** (1946), pp. 122–146.

[19] Parker, D. S. and C. Delobel, *Algorithmic applications for a new result on multivalued dependencies*, in: *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, 1979, pp. 67–74.

[20] Sagiv, Y., *An algorithm for inferring multivalued dependencies with an application to propositional logic*, Journal of the ACM **27** (1980), pp. 250–262.

[21] Sagiv, Y., C. Delobel, D. S. Parker Jr. and R. Fagin, *An equivalence between relational database dependencies and a fragment of propositional logic*, Journal of the ACM **28** (1981), pp. 435–453.

[22] Suciu, D., *On database theory and XML*, SIGMOD Record **30** (2001), pp. 39–45.

[23] Vianu, V., *A web odyssey: from Codd to XML*, in: *Principles of Database Systems* (2001), pp. 1–15.