

Characterising nested database dependencies by fragments of propositional logic

Sven Hartmann, Sebastian Link*

Information Science Research Centre, Massey University, New Zealand

Abstract

We extend the earlier results on the equivalence between the Boolean and the multivalued dependencies in relational databases and fragments of the Boolean propositional logic. It is shown that these equivalences are still valid for the databases that store complex data elements obtained from the recursive nesting of record, list, set and multiset constructors. The major proof argument utilises properties of Brouwerian algebras.

The equivalences have several consequences. Firstly, they provide new insights into databases that are not in first normal form. Secondly, they characterise the implication of data dependencies in nested databases in purely logical terms. The database designer can take advantage of these equivalences to reduce database design problems to well-studied problems in Boolean propositional logic. Furthermore, relational database design solutions can be reused to solve problems for nested databases.

© 2007 Elsevier B.V. All rights reserved.

MSC: 68P15

Keywords: Logic in databases; Nested databases; Boolean dependency; Multivalued dependency; Brouwerian algebra; Propositional logic

1. Introduction

Functional dependencies (FDs, [13]) and multivalued dependencies (MVDs, [14,18]) 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 into two of their projections without loss of information. According to [15] 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 [30,31]. 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 [28,29]. In particular, the equivalence reduces to the fragment of Horn clauses when FDs are studied by themselves [17]. The next example illustrates Fagin's finding that a two-tuple counterexample relation for the implication of an FD φ by a set Σ of FDs

* Corresponding author.

E-mail addresses: s.hartmann@massey.ac.nz (S. Hartmann), s.link@massey.ac.nz (S. Link).

allows one to define a truth assignment that satisfies all the Horn clauses that correspond to the FDs in Σ but violates at least one of the Horn clauses that correspond to φ , and vice versa. Such a truth assignment assigns *true* to precisely those variables that correspond to attributes on which the two tuples of the counterexample relation agree.

Example 1. Consider the relation schema

$$\text{LECTURE} = \{\text{Class, Lecturer, Time, Room}\}$$

together with the following set Σ of functional dependencies on LECTURE:

- Class \rightarrow Lecturer,
- Class, Time \rightarrow Room,
- Lecturer, Time \rightarrow Class, and
- Room, Time \rightarrow Class.

Suppose we would like to find out whether Room and Time together form a superkey for LECTURE. That is, the functional dependency φ :

$$\text{Room, Time} \rightarrow \text{Class, Lecturer}$$

is implied by Σ . This decision problem is equivalent to the problem of deciding whether both of the following propositional Horn clauses,¹ represented in implicational form,

$$(\text{Room} \wedge \text{Time}) \Rightarrow \text{Class} \quad \text{and} \quad (\text{Room} \wedge \text{Time}) \Rightarrow \text{Lecturer}$$

are logically implied by the following set Σ' of propositional Horn clauses:

- Class \Rightarrow Lecturer,
- (Class \wedge Time) \Rightarrow Room,
- (Lecturer \wedge Time) \Rightarrow Class,
- (Room \wedge Time) \Rightarrow Class.

Furthermore, the functional dependency

$$\text{Class, Lecturer, Room} \rightarrow \text{Time}$$

is not implied by Σ . A counterexample to this implication is given, for instance, by the following two-tuple relation r :

$$t_1 = (\text{Databases, H. Simpson, 2:30 pm, 3.12}), \quad \text{and} \\ t_2 = (\text{Databases, H. Simpson, 4:30 pm, 3.12}).$$

The truth assignment that assigns *true* to the propositional variables Class, Lecturer and Room, and *false* to the variable Time makes all Horn clauses in Σ' *true* but leaves the Horn clause

$$(\text{Class} \wedge \text{Lecturer} \wedge \text{Room}) \Rightarrow \text{Time}$$

false. \square

Many researchers have remarked that classical database design problems need to be revisited in new data formats [34,36]. Biskup [9] 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 [23,24,27]. The Entity-Relationship model and its extensions require record, set and (disjoint) union constructor [12,35]. A minimal set of type constructors supported by any object-oriented data model includes record, list, set and multiset (bag) constructor [5,7,32]. Genomic sequence

¹ The attributes of LECTURE are now used as propositional variables.

data models call for support of records, lists and sets [11,25,33]. Finally, XML requires at least record (concatenation), list (Kleene Closure), union (optionality), and reference constructor [1,10].

The major goal of this paper is to generalise the well-known equivalence results [30,31] from relational databases to nested databases that support record, list, set and multiset constructor. It is our intention not to focus on the specifics of any particular data model in order to place emphasis on the impact of the type constructors themselves. 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, list, set and multiset constructor. This approach leads to Brouwerian algebras [26] 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.

The following example, taken from the field of image processing, illustrates a typical scenario in which databases store complex values (e.g., bit sequences). It shows how functional and multivalued dependencies can be used in the presence of record and list constructor. We will use this example throughout the paper.

Example 2. Digital halftoning is an application of the matrix rounding problem [4]. The problem is to convert a continuous-tone image into a binary one that looks similar. The input matrix A represents a digital (gray) image, where a_{ij} represents the brightness level of the (i, j) -pixel in the $n \times n$ pixel grid. Typically, n is between 256 and 4096, and a_{ij} is an integral multiple of $\frac{1}{256}$: this means that we use 256 brightness levels. If we want to send an image using fax or print it out by a dot or ink-jet printer, brightness levels available are limited. Instead, we replace the input matrix A by an integral matrix B so that each pixel uses only two brightness levels. Here, it is important that B looks similar to A ; in other words, B should be an approximation of A . In this sense, an approximation of input matrix A is a $\{0,1\}$ -matrix B that minimises the distance $|\sum_{(i,j) \in R} a_{i,j} - \sum_{(i,j) \in R} b_{i,j}|$ for all $R \in \mathcal{R}$. In this formula \mathcal{R} denotes the set of regions of neighbours, for instance the set of all pairs of indices that denote 2×1 , 1×2 and 2×2 submatrices. A region has therefore one of the following forms

$$\begin{array}{|c|c|} \hline a & b \\ \hline c & d \\ \hline \end{array} \quad \begin{array}{|c|c|} \hline a & b \\ \hline \end{array} \quad \begin{array}{|c|} \hline a \\ \hline b \\ \hline \end{array},$$

and can be represented as a list of either two or four elements. The regions may have all different kinds of shapes in practice. In order to make the example more illustrative, we assume from now on that the input matrix has entries in $\{0, \frac{1}{2}, 1\}$, i.e., uses three brightness levels. Input regions can be best approximated by a number of different output regions. All inputs with overall brightness $\frac{1}{2}$ and length two, i.e. $[0, \frac{1}{2}]$ or $[\frac{1}{2}, 0]$, could be mapped to any of $[0, 1]$, $[1, 0]$ or $[0, 0]$, each of which has distance $\frac{1}{2}$. In this sense, the set of input sequences ($\{[0, \frac{1}{2}], [\frac{1}{2}, 0]\}$) is determined by the overall brightness of the input sequences ($\frac{1}{2}$) and the length of the input sequence (2), independently from the set of output sequences ($\{[0, 1], [1, 0], [0, 0]\}$). This is true for any inputs and outputs, e.g., all inputs with overall brightness $\frac{3}{2}$ and length four, such as $[0, 0, 1, \frac{1}{2}]$, can be mapped to any of $[0, 0, 0, 1]$, $[0, 0, 1, 0]$, $[0, 1, 0, 0]$, $[1, 0, 0, 0]$, $[0, 0, 1, 1]$, $[0, 1, 0, 1]$, $[1, 0, 0, 1]$, $[0, 1, 1, 0]$, $[1, 0, 1, 0]$, $[1, 1, 0, 0]$.

Consider a database which stores input and output sequences together with the overall brightness of the input sequence. A schema for such a database may be HALFTONING(Brightness, INPUT[Level], OUTPUT[Bit]). It is then desirable to find a $\{0,1\}$ -matrix B that has for every of the possible regions of input matrix A a corresponding output region that are stored together as an entry in the database. The input matrix $A = \begin{pmatrix} 0 & 0 \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix}$ has for instance the

approximation $B = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$. Every 2×2 matrix has five input sequences and the mappings that produce B from A are as follows: $[0, 0] \mapsto [0, 0]$, $[\frac{1}{2}, \frac{1}{2}] \mapsto [0, 1]$, $[0, \frac{1}{2}] \mapsto [0, 0]$ (left column), $[0, \frac{1}{2}] \mapsto [0, 1]$ (right column) and $[0, 0, \frac{1}{2}, \frac{1}{2}] \mapsto [0, 0, 0, 1]$.

The matrix $\begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$, however, is not an approximation of A as the sequence $[\frac{1}{2}, \frac{1}{2}]$ should not be mapped to $[0, 0]$.

Constraints that a database designer may choose to specify for this application are the following:

- (1) The length of the input sequence determines the length of the output sequence, and vice versa.
- (2) The overall brightness and length of the input sequence together determine the set of all input sequences independently from the set of the output sequences.

It appears that the English description of these constraints points us to a functional dependency in the first case and a multivalued dependency in the second case. \square

Our aim is to describe the implication of dependency classes in the presence of type constructors in purely logical terms. The next example illustrates that the presence of type constructors causes difficulties in extending the results from the relational theory. While attributes (i.e. the join-irreducible elements of a relation schema) are sufficient for defining functional and multivalued dependencies in the relational model of data the join-irreducibles of a nested database schema turn out to be insufficient for this purpose. We demonstrate this fact by the following example which will also be used throughout the article.

Example 3. Consider a simple example of a purchase profile that supermarkets and Online shops may utilise. A single entry consists of the name of the customer, a bag of items the customer bought, and the discount of this purchase received by the customer. Moreover, every item of the customer's bag consists of an article together with the price of that article. A database schema for such an application may look as follows

PROFILE(Customer, BAG(ITEM(Article, Price)), Discount).

An actual entry in the database may be

(Homer, ((Chocolate, 3\$), (Chocolate, 3\$), (Beer, 4\$), (Beer, 5\$)), 2\$).

Suppose that Homer received his discount of 2\$ since beer that costs 4\$ or more is on special. Intuitively, customers with the same bag of items (i.e. their tuples agree on BAG(ITEM(Article, Price))) should receive the same discount (i.e. their tuples also agree on Discount). This is an actual functional dependency that involves the equality of complex data objects, in this case a multiset. The presence of the multiset constructor shows some surprises. Consider for instance a second data element

(Bart, ((Chocolate, 4\$), (Chocolate, 5\$), (Beer, 3\$), (Beer, 3\$)), 0\$).

Bart bought the same bag of articles and has the same bag of prices as Homer (i.e. both Chocolate and Beer occur twice in both bags, 3\$ occurs twice in both bags and 4\$ and 5\$ both occur once in both bags) yet did not receive any discount. This is consistent with respect to the functional dependency since Bart did not have the same bag of items as Homer. In order to receive a discount it matters which articles are bought to which price. In order to specify such a functional dependency one requires the nested attribute BAG(ITEM(Article, Price)) which is not join-irreducible (refer to the end of Section 2 for a formal definition). \square

Example 3 illustrates the necessity of studying the question which elements, in addition to the join-irreducibles, are required to specify dependencies in the presence of certain type constructors. However, the extension of the results from the relational model of data [17,30,31] is problematic for other reasons as well. Unlike the relational model of data where attributes are incomparable with respect to set inclusion the join-irreducible elements of a nested database schema do no longer form an anti-chain. However, the non-trivial structure of the join-irreducibles can be encoded using the Horn clauses. Furthermore, double complementation is no longer an involution in Brouwerian algebras which complicates proof arguments, especially in the case of multivalued dependencies. In the relational model of data the implication of Boolean and multivalued dependencies over unrestricted relations is equivalent to the implication over two-tuple relations. The original proof of the Equivalence theorems is based on a particular truth assignment that interprets precisely those attribute names as *true* on which the two tuples agree. Given an arbitrary truth assignment it is straightforward to find two tuples which precisely agree on those attributes which are interpreted as *true*. However, the existence (and construction) of such a two-element instance is far from being obvious in nested databases. The constructive existence proof requires a detailed analysis of each individual type constructor [21]. In the current paper we will use this result to establish equivalences between database dependencies in nested databases and fragments of propositional logic. Therefore, it turns out that the earlier results can be extended to a more general algebraic framework.

The paper is organised as follows. We will describe our data model in Section 2. We will use Section 3 to analyse which information is required to identify arbitrary nested data elements in the presence of different type constructors. It turns out that join-irreducibles are sufficient when dealing with arbitrary finite nesting of records and lists, but insufficient as soon as set or multiset constructor are utilised, as already pointed out by Example 3. In Section 4 we

first introduce the class of functional and multivalued dependencies in the presence of record and list constructor. Subsequently, we extend functional dependencies to the general case in which also set and multiset constructor are allowed. Finally, we extend functional dependencies to arbitrary Boolean dependencies. The equivalence results are presented in Section 5. In the presence of records and lists only, FDs still correspond to Boolean–Horn clauses while MVDs correspond to implicational statements with conjunctions of variables as antecedent and disjunctions over conjunctions of variables in the consequent. In both cases the structure of join-irreducibles needs to be encoded as Horn clauses. If set and multiset constructor are also considered then Boolean dependencies correspond to Boolean propositional formulae. In this case an extension of the set of join-irreducible elements is required and the structure of these extended join-irreducibles is also encoded by the Horn clauses. In Section 6 we use the equivalence results to apply relational database design solutions to nested database design problems. In particular, we determine upper bounds for the time-complexity of the associated implication problems.

2. Nested attributes

The goal of this section is to provide a unifying framework for the study of dependency classes in the presence of complex type constructors. Therefore, we introduce a data model based on the nesting of attributes and subtyping. Nested data models have been proposed to overcome severe limitations of the relational data model when designing many practical database applications [2].

2.1. Database schemata

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*. Flat attributes will be denoted by upper-case characters from the start of the alphabet such as A, B, C etc.

In the following we will 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., $\mathcal{U} \cap \mathcal{L} = \emptyset$.

Database schemata in our data model will be given in the form of nested attributes. Let \mathcal{U} be a universe and \mathcal{L} a set of labels. The set $\mathcal{N}(\mathcal{U}, \mathcal{L})$ of *nested attributes over \mathcal{U} and \mathcal{L}* is the smallest set satisfying the following conditions:

- (1) $\lambda \in \mathcal{N}(\mathcal{U}, \mathcal{L})$,
- (2) $\mathcal{U} \subseteq \mathcal{N}(\mathcal{U}, \mathcal{L})$,
- (3) for $L \in \mathcal{L}$ and $N_1, \dots, N_k \in \mathcal{N}(\mathcal{U}, \mathcal{L})$ with $k \geq 1$ we have $L(N_1, \dots, N_k) \in \mathcal{N}(\mathcal{U}, \mathcal{L})$,
- (4) for $L \in \mathcal{L}$ and $N \in \mathcal{N}(\mathcal{U}, \mathcal{L})$ we have $L[N], L\{N\}, L\langle N \rangle \in \mathcal{N}(\mathcal{U}, \mathcal{L})$.

We call λ the *null attribute*, $L(N_1, \dots, N_k)$ *record-valued attribute*, $L[N]$ *list-valued attribute*, $L\{N\}$ *set-valued attribute* and $L\langle N \rangle$ *multiset-valued attribute*. From now on, we assume that a set \mathcal{U} of attribute names, and a set \mathcal{L} of labels is fixed, and write \mathcal{N} instead of $\mathcal{N}(\mathcal{U}, \mathcal{L})$. Let \mathcal{T} be a set of type constructors, in our case any subset of {record, list, set, multiset}. We use $\mathcal{N}_{\mathcal{T}}$ to denote the set of all nested attributes in \mathcal{N} that are generated from flat attributes by finitely many recursive applications of type constructors in \mathcal{T} . We have $\mathcal{N}_{\mathcal{T}} \subseteq \mathcal{N}_{\mathcal{T}'}$ whenever $\mathcal{T} \subseteq \mathcal{T}'$. We write \mathcal{N} instead of $\mathcal{N}_{\{\text{record, list, set, multiset}\}}$.

The next example illustrates how to formally generate the nested attributes that have already occurred in the examples of the introduction.

Example 4. Suppose we are given flat attributes such as Brightness, Level, and Bit, and labels such as HALFTONING, INPUT, and OUTPUT. We may then generate the list-valued attributes INPUT[Level] and OUTPUT[Bit], as well as the record-valued attribute

HALFTONING(Brightness, INPUT[Level], OUTPUT[Bit]).

Given flat attributes such as Customer, Article, Price and Discount, as well as labels PROFILE, ITEM and BAG, we may generate a record-valued attribute ITEM(Article, Price), a multiset-valued attribute BAG(Item(Article, Price)), and a record-valued attribute

PROFILE(Customer, BAG⟨ITEM(Article, Price)⟩, Discount).

Using the null attribute λ we may generate nested attributes such as INPUT[λ], HALFTONING(Brightness, INPUT[λ], λ), BAG⟨ITEM(λ , λ)⟩ or PROFILE(Customer, BAG⟨ITEM(λ , λ)⟩, Discount). \square

We can extend the mapping dom from flat attributes to nested attributes, i.e., we define a set $dom(N)$ of possible data elements for every nested attribute $N \in \mathcal{N}$. For a nested attribute $N \in \mathcal{N}$ we define the *domain* $dom(N)$ as follows: $dom(\lambda) = \{\text{ok}\}$, $dom(A)$ for $A \in \mathcal{U}$ as before, $dom(L(N_1, \dots, N_k)) = \{(v_1, \dots, v_k) \mid v_i \in dom(N_i) \text{ for } i = 1, \dots, k\}$, i.e., the set of all k -tuples (v_1, \dots, v_k) with $v_i \in dom(N_i)$ for all $i = 1, \dots, k$, $dom(L\{N\}) = \{\{v_1, \dots, v_n\} \mid v_i \in dom(N) \text{ for } i = 1, \dots, n\}$, i.e., the set of all finite sets with elements in $dom(N)$, $dom(L\langle N \rangle) = \{\{v_1, \dots, v_n\} \mid v_i \in dom(N) \text{ for } i = 1, \dots, n\}$, i.e., the set of all finite multisets with elements in $dom(N)$, and $dom(L[N]) = \{[v_1, \dots, v_n] \mid v_i \in dom(N) \text{ for } i = 1, \dots, n\}$, i.e., the set of all finite lists with elements in $dom(N)$. The empty set, multiset and list are denoted by \emptyset , $\langle \rangle$, and $[\]$, respectively. The value ok can be interpreted as the null value “some information exists, but is currently omitted”.

Next we give some examples of nested tuples from the domains of the nested attributes in [Example 4](#).

Example 5. The data element

$$t = (\text{Homer}, \langle (\text{Chocolate}, 3\$), (\text{Chocolate}, 3\$), (\text{Beer}, 4\$), (\text{Beer}, 5\$), 2\$ \rangle)$$

is from the domain of

$$N = \text{PROFILE}(\text{Customer}, \text{BAG}\langle \text{ITEM}(\text{Article}, \text{Price}) \rangle, \text{Discount}).$$

Moreover, the data element

$$t' = (\text{Homer}, \langle (\text{ok}, \text{ok}), (\text{ok}, \text{ok}), (\text{ok}, \text{ok}), (\text{ok}, \text{ok}), 2\$ \rangle)$$

is from the domain of

$$X = \text{PROFILE}(\text{Customer}, \text{BAG}\langle \text{ITEM}(\lambda, \lambda) \rangle, \text{Discount}).$$

Notice that knowledge of N and t' still reveals that Homer has bought four items. \square

2.2. Subschemas

The replacement of attribute names by the null attribute λ within a nested attribute decreases the amount of information that is modelled by the corresponding schemata. This observation results in the definition of an order between database schemata.

The *subattribute relation* \leq on the set of nested attributes \mathcal{N} over \mathcal{U} and \mathcal{L} is defined by the following rules, and the following rules only:

- $N \leq N$,
- $\lambda \leq A$ for all flat attributes $A \in \mathcal{U}$,
- $\lambda \leq N$ for all list-valued, set-valued and multiset-valued attributes N ,
- $L(N_1, \dots, N_k) \leq L(M_1, \dots, M_k)$ whenever $N_i \leq M_i$ for all $i = 1, \dots, k$, and
- $L[N] \leq L[M]$, $L\{N\} \leq L\{M\}$, $L\langle N \rangle \leq L\langle M \rangle$ whenever $N \leq M$.

For N, M we say that M is a *subattribute* of N if and only if $M \leq N$ holds. We write $M \not\leq N$ if M is not a subattribute of N , and $M < N$ in case $M \leq N$ and $M \neq N$.

Lemma 6. *The subattribute relation is a partial order on nested attributes.* \square

The subattribute relationship between nested attributes generalises the inclusion relationship between sets of attributes in the relational data model. Informally, M is a subattribute of N 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 $\pi_M^N : dom(N) \rightarrow dom(M)$ in case $M \leq N$ holds. For $M \leq N$ this *projection function* $\pi_M^N : dom(N) \rightarrow dom(M)$ is defined as follows:

- if $N = M$, then $\pi_M^N = id_{dom(N)}$ is the identity on $dom(N)$,
- if $M = \lambda$, then $\pi_\lambda^N : dom(N) \rightarrow \{\text{ok}\}$ is the constant function that maps $v \in dom(N)$ to ok ,

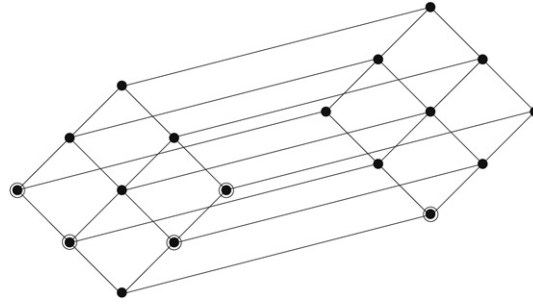


Fig. 1. Brouwerian algebra of HALFTONING(Brightness, INPUT[Level], OUTPUT[Bit]).

- if $N = L(N_1, \dots, N_k)$ and $M = L(M_1, \dots, M_k)$, then $\pi_M^N = \pi_{M_1}^{N_1} \times \dots \times \pi_{M_k}^{N_k}$ maps the tuple $(v_1, \dots, v_k) \in \text{dom}(N)$ to $(\pi_{M_1}^{N_1}(v_1), \dots, \pi_{M_k}^{N_k}(v_k)) \in \text{dom}(M)$,
- if $N = L[N']$ and $M = L[M']$, then $\pi_M^N : \text{dom}(N) \rightarrow \text{dom}(M)$ maps the list $[v_1, \dots, v_n] \in \text{dom}(N)$ to $[\pi_{M'}^{N'}(v_1), \dots, \pi_{M'}^{N'}(v_n)] \in \text{dom}(M)$,
- if $N = L\{N'\}$ and $M = L\{M'\}$, then $\pi_M^N : \text{dom}(N) \rightarrow \text{dom}(M)$ maps the set $\{v_1, \dots, v_n\} \in \text{dom}(N)$ to $\{\pi_{M'}^{N'}(v_1), \dots, \pi_{M'}^{N'}(v_n)\} \in \text{dom}(M)$, and
- if $N = L\langle N'\rangle$ and $M = L\langle M'\rangle$, then $\pi_M^N : \text{dom}(N) \rightarrow \text{dom}(M)$ maps the multiset $\langle v_1, \dots, v_n \rangle \in \text{dom}(N)$ to $\langle \pi_{M'}^{N'}(v_1), \dots, \pi_{M'}^{N'}(v_n) \rangle \in \text{dom}(M)$.

The projection function tells us precisely how to map a database instance of some schema to an instance of any of its subschemata.

Example 7. Recall the definition of N , X , t and t' from Example 5. Using the projection function π we can see that $t' = \pi_X^N(t)$, i.e., t' results from t by masking out the information on Articles and Prices. \square

2.3. Brouwerian algebra of subattributes

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 partial order \subseteq , set union \cup , set intersection \cap and set difference $-$. We will now extend these operations to nested attributes. The inclusion order \subseteq has already been generalised by the subattribute relationship \leq . The set $\text{Sub}(N)$ of *subattributes* of N is $\text{Sub}(N) = \{M \mid M \leq N\}$.

We study the algebraic structure of the poset $(\text{Sub}(N), \leq)$. A *Brouwerian algebra* [26] is a lattice $(L, \sqsubseteq, \sqcup, \sqcap, \dashv, 1)$ with top element 1 and a binary operation \dashv which satisfies $a \dashv b \sqsubseteq c$ iff $a \sqsubseteq b \sqcup c$ for all $c \in L$. In this case, the operation \dashv is called the *pseudo-difference*. The *Brouwerian complement* $\neg a$ of $a \in L$ is then defined by $\neg a = 1 \dashv a$. A Brouwerian algebra is also called a co-Heyting algebra or a dual Heyting algebra. The system of all closed subsets of a topological space is a well-known Brouwerian algebra, see [26]. The definition of the subattribute relationship \leq completely determines the operations of join, meet and pseudo-difference. 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 relational data model [22].

Theorem 8. $(\text{Sub}(N), \leq, \sqcup_N, \sqcap_N, \dashv_N, N)$ forms a Brouwerian algebra for every $N \in \mathcal{N}$. \square

The nested attribute N is the top element of $(\text{Sub}(N), \leq)$. The *bottom element* λ_N of $\text{Sub}(N)$ is given by $\lambda_N = L(\lambda_{N_1}, \dots, \lambda_{N_k})$ whenever $N = L(N_1, \dots, N_k)$, and $\lambda_N = \lambda$ whenever N is not a record-valued attribute. The Brouwerian algebra for HALFTONING(Brightness, INPUT[Level], OUTPUT[Bit]) is illustrated in Fig. 1.

If the context allows, we omit the index N from the operations $\sqcup_N, \sqcap_N, \dashv_N$ and from λ_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 $a = c$. The set of join-irreducible elements of $(\text{Sub}(N), \leq, \sqcup, \sqcap, \lambda_N)$ is denoted by $\mathcal{J}(N)$. We refer to elements of $\mathcal{J}(N)$ as join-irreducible elements of N . For instance, the join-irreducibles of HALFTONING(Brightness, INPUT[Level], OUTPUT[Bit]) are circled in Fig. 1.

3. Identifying nested data elements

We have seen in [Example 3](#) of the introduction that nested data elements $t \in \text{dom}(N)$ cannot be uniquely identified by their projections on the join-irreducibles $X \in \mathcal{J}(N)$. This implies that join-irreducibles are insufficient to specify semantic constraints on nested database schemata. In order to derive equivalences between the implication of data dependencies and fragments of propositional logic it is essential to clarify which other subattributes apart from the join-irreducibles need to be interpreted as propositional variables. Moreover, the minimal number of subattributes necessary to identify the nested data elements will provide us with the right measure for studying the time-complexity of the implication problems for the classes of dependencies.

Let N be an arbitrary nested attribute composed of type constructors in \mathcal{T} only, i.e., $N \in \mathcal{N}_{\mathcal{T}}$. Which set of subattributes of N allows to uniquely identify any nested data element in $\text{dom}(N)$? In other words, what is the minimal set $\mathfrak{A}_{\mathcal{T}}(N) \subseteq \text{Sub}(N)$ such that every element $t \in \text{dom}(N)$ is uniquely determined by its projections $\{\pi_X^N(t) \mid X \in \mathfrak{A}_{\mathcal{T}}(N)\}$, i.e., for all $t, t' \in \text{dom}(N)$ the following holds: if $\pi_X^N(t) = \pi_X^N(t')$ for all $X \in \mathfrak{A}_{\mathcal{T}}(N)$, then $t = t'$?

3.1. Relational databases

Consider the simple case where \mathcal{T} consists of the record constructor only. Suppose further that nested attributes are generated from flat attributes by a single application of the record constructor. That is, $N = R(A_1, \dots, A_k)$ for flat attributes A_1, \dots, A_k and a label R . This is just a different notation for the relation schema $R = \{A_1, \dots, A_k\}$. Now, every tuple t over R (i.e. every function $t : R \rightarrow \bigcup_{i=1}^k \text{dom}(A_i)$ with $t(A_i) \in \text{dom}(A_i)$ for $i = 1, \dots, k$) is completely determined by its projections $\{t(A_1), \dots, t(A_k)\}$. In fact, in order to store a tuple we store its values on the individual attributes. This is just the representation of a relational database as a table in which attributes are used as column names. Let us view the relation schema R as the nested attribute N . In this case every $t \in \text{dom}(N)$ is uniquely determined by its projections

$$\{\pi_{L(A_1, \lambda, \dots, \lambda)}^N(t), \dots, \pi_{L(\lambda, \dots, \lambda, A_k)}^N(t)\}.$$

From an algebraic point of view the k subattributes

$$L(A_1, \lambda, \dots, \lambda), \dots, L(\lambda, \dots, \lambda, A_k)$$

are the join-irreducible elements of $(\text{Sub}(N), \leq, \sqcup, \sqcap, \lambda_N)$.

3.2. Including lists

Consider now the case where $N \in \mathcal{N}_{\{\text{record}, \text{list}\}}$. In this case the following lemma is fundamental [22].

Lemma 9. *Let $X, Y \in \text{Sub}(N)$ and $t_1, t_2 \in \text{dom}(N)$. If $\pi_X^N(t_1) = \pi_X^N(t_2)$ and $\pi_Y^N(t_1) = \pi_Y^N(t_2)$, then $\pi_{X \sqcup Y}^N(t_1) = \pi_{X \sqcup Y}^N(t_2)$. \square*

That is, the two projections of a tuple on two subattributes X and Y uniquely determine the projection of that tuple on the join $X \sqcup Y$. This shows, in particular, that $\mathfrak{A}_{\mathcal{T}}(N)$ is still the set of join-irreducible elements of N . Therefore, the presence of the list constructor does not change the set of subattributes necessary to identify the nested data elements.

3.3. The general case

If we add set or multiset constructor to \mathcal{T} , then it becomes insufficient to consider join-irreducible elements. In fact, there is some nested attribute $N \in \mathcal{N}$ and distinct elements of $\text{dom}(N)$ which agree on all projections to join-irreducibles of N . An example of such a nested attribute and two nested data elements is $\text{BAG}(\text{ITEM}(\text{Article}, \text{Price}))$ and the two tuples

$$\langle (\text{Chocolate}, 3\$), (\text{Chocolate}, 3\$), (\text{Beer}, 4\$), (\text{Beer}, 5\$) \rangle \quad \text{and} \\ \langle (\text{Chocolate}, 4\$), (\text{Chocolate}, 5\$), (\text{Beer}, 3\$), (\text{Beer}, 3\$) \rangle.$$

In the presence of set and multiset constructor we face the difficulty of characterising those pairs of subattributes X and Y of N for which $t_1, t_2 \in \text{dom}(N)$ exist such that $\pi_X^N(t_1) = \pi_X^N(t_2)$, $\pi_Y^N(t_1) = \pi_Y^N(t_2)$ and $\pi_{X \sqcup Y}^N(t_1) \neq \pi_{X \sqcup Y}^N(t_2)$.

In other words, what subattributes of N (other than the join-irreducibles) are necessary to uniquely identify any nested data element over N ? The following definition is used to answer this question.

Definition 10. Let $N \in \mathcal{N}$. The subattributes $X, Y \in \text{Sub}(N)$ are *reconcilable* if and only if one of the following conditions is satisfied

- (1) $Y \leq X$ or $X \leq Y$,
- (2) $N = L(N_1, \dots, N_k)$, $X = L(X_1, \dots, X_k)$, $Y = L(Y_1, \dots, Y_k)$ where X_i and Y_i are reconcilable for all $i = 1, \dots, k$, or
- (3) $N = L[N']$, $X = L[X']$, $Y = L[Y']$ where X' and Y' are reconcilable. \square

The two subattributes $\text{BAG}(\text{ITEM}(\text{Article}, \lambda))$ and $\text{BAG}(\text{ITEM}(\lambda, \text{Price}))$ of $\text{BAG}(\text{ITEM}(\text{Article}, \text{Price}))$ are not reconcilable.

The following two lemmata were proven as Lemma 21 and Lemma 31, respectively, in [21]. Recall that an ideal [3] of some poset (S, \leq) is a subset $\mathcal{I} \subseteq S$ which is closed downwards with respect to \leq , i.e., if $X \in \mathcal{I}$ and $Y \leq X$, then $Y \in \mathcal{I}$ as well.

Lemma 11. Let $N \in \mathcal{N}$, and $\emptyset \neq \mathcal{X} \subseteq \text{Sub}(N)$ an ideal with respect to \leq with the property that for reconcilable $X, Y \in \mathcal{X}$ also $X \sqcup Y \in \mathcal{X}$ holds. Then there are $t, t' \in \text{dom}(N)$ such that for all $W \in \text{Sub}(N)$ we have $\pi_W^N(t) = \pi_W^N(t')$ if and only if $W \in \mathcal{X}$. \square

In contrast to the relational model of data the existence of $t, t' \in \text{dom}(N)$ in Lemma 11 is far from being obvious. The proof in [21] is constructive and consists of a careful analysis of the type constructors.

Lemma 12. Let $N \in \mathcal{N}$ and $X, Y \in \text{Sub}(N)$. Then $\mathcal{X} = \{U \sqcup V : U \leq X, V \leq Y, U \text{ and } V \text{ are reconcilable}\}$ is a non-empty ideal with respect to \leq and for all $S, T \in \mathcal{X}$ that are reconcilable follows $S \sqcup T \in \mathcal{X}$. \square

Theorem 13. Let $N \in \mathcal{N}$. For all $X, Y \in \text{Sub}(N)$ we have that X and Y are reconcilable if and only if for all $t, t' \in \text{dom}(N)$ with $\pi_X^N(t) = \pi_X^N(t')$ and $\pi_Y^N(t) = \pi_Y^N(t')$ also $\pi_{X \sqcup Y}^N(t) = \pi_{X \sqcup Y}^N(t')$ holds.

Proof. The sufficiency of reconcilability was proven in [21, Lemma 16]. In order to prove the necessity of reconcilability we assume that $X, Y \in \text{Sub}(N)$ are not reconcilable. We then need to show that there are $t, t' \in \text{dom}(N)$ such that $\pi_X^N(t) = \pi_X^N(t')$, $\pi_Y^N(t) = \pi_Y^N(t')$ but $\pi_{X \sqcup Y}^N(t) \neq \pi_{X \sqcup Y}^N(t')$. According to Lemma 11 it remains to find an ideal \mathcal{X} that is closed under the join of non-reconcilable elements and where $X, Y \in \mathcal{X}$ and $X \sqcup Y \notin \mathcal{X}$ holds. However, such an ideal \mathcal{X} is given in Lemma 12. \square

It still remains to clarify which projections are necessary and sufficient to identify every nested data element over a nested attribute generated by finitely many applications of record, list, set and multiset constructor.

Definition 14. Let $N \in \mathcal{N}$. The set of *extended join-irreducibles* of N is the smallest set $\mathcal{E}(N) \subseteq \text{Sub}(N)$ with the properties that $\mathcal{J}(N) \subseteq \mathcal{E}(N)$, and that for all $X, Y \in \mathcal{E}(N)$ which are not reconcilable also $X \sqcup Y \in \mathcal{E}(N)$ holds. \square

The extended join-irreducibles of N form therefore the smallest set that contains the join-irreducibles of N and that is closed under the join of subattributes that are not reconcilable. In the absence of sets and multisets we have $\mathcal{E}(N) = \mathcal{J}(N)$ since every pair of subattributes is reconcilable. If N is a set- or multiset-valued attribute, then $\mathcal{E}(N) = \text{Sub}(N)$. If \mathcal{T} consists of records, lists, sets and multisets, then $\mathcal{A}_{\mathcal{T}}(N) = \mathcal{E}(N)$. This seems now very natural: for any $X, Y \in \mathcal{E}(N)$ for which the two projections $\pi_X^N(t)$ and $\pi_Y^N(t)$ of some $t \in \text{dom}(N)$ do not uniquely determine the value of $\pi_{X \sqcup Y}^N(t)$, the subattributes X and Y cannot be reconcilable, and $X \sqcup Y$ is therefore included in $\mathcal{E}(N)$.

3.4. Characterising reconcilability

This subsection introduces the notion of a *unit* which we apply in the proof of the equivalence results.

There is a relatively simple way to reduce the notion of reconcilability to the notion of comparability with respect to \leq . The idea is to choose the units U of N such that for all subattributes $V, W \in \text{Sub}(N)$ we have that V and W are reconcilable if and only if $V \sqcap U$ and $W \sqcap U$ are comparable with respect to \leq for all units U of N . Recall that two subattributes X and Y are comparable with respect to \leq if and only if $X \leq Y$ or $Y \leq X$ holds. This motivates the following definition.

Definition 15. Let $N \in \mathcal{N}$. A nested attribute $U \in \mathcal{N}$ is a *unit* of N if and only if

- (1) $U \in \text{Sub}(N)$, and
- (2) $\forall X, Y \leq U$ if X and Y are reconcilable, then $X \leq Y$ or $Y \leq X$, and
- (3) U is \leq -maximal with the properties (1) and (2), i.e., every U' that satisfies (1) and (2) is not a proper superattribute of U (i.e. it is not the case that $U < U'$ holds).

The set of all units of N is denoted by $\mathcal{U}(N)$. \square

Example 16. The units of

PROFILE(Customer, BAG(ITEM(Article, Price)), Discount)

are

$U_1 = \text{PROFILE}(\text{Customer}, \lambda, \lambda)$,

$U_2 = \text{PROFILE}(\lambda, \text{BAG}\langle \text{ITEM}(\text{Article}, \text{Price}) \rangle, \lambda)$, and

$U_3 = \text{PROFILE}(\lambda, \lambda, \text{Discount})$.

The two subattributes U_1 and U_3 are reconcilable. In fact, for every $U \in \{U_1, U_2, U_3\}$ we have $U_1 \sqcap U \leq U_3 \sqcap U$ or $U_3 \sqcap U \leq U_1 \sqcap U$. However, the subattributes

$X = \text{PROFILE}(\lambda, \text{BAG}\langle \text{ITEM}(\text{Article}, \lambda) \rangle, \lambda)$, and

$Y = \text{PROFILE}(\lambda, \text{BAG}\langle \text{ITEM}(\lambda, \text{Price}) \rangle, \lambda)$

are not reconcilable. In fact, $X = X \sqcap U_2$ and $Y = Y \sqcap U_2$ are incomparable with respect to \leq . \square

4. Data dependencies

In this section we repeat the definitions of functional and multivalued dependencies in the presence of record and list constructor [22], as well as the definition of functional dependencies in the presence of record, list, set and multiset constructor [21]. Moreover, we introduce the class of *Boolean dependencies* on nested attributes generated by record, list, set and multiset constructor, and formalise the notion of implication for these classes of dependencies.

4.1. FDs and MVDs in the presence of lists

Our framework enables us to easily formalise the following intuitive idea of a functional dependency. Namely that a set of nested tuples satisfies an FD $X \rightarrow Y$ if every pair of tuples with the same projection on X also has the same projection on Y .

Definition 17. Let $N \in \mathcal{N}_{\{\text{record}, \text{list}\}}$ be a nested attribute. A *functional dependency on N* is an expression of the form $X \rightarrow Y$ where $X, Y \in \text{Sub}(N)$. A set $r \subseteq \text{dom}(N)$ satisfies the functional dependency $X \rightarrow Y$ on N , denoted by $\models_r X \rightarrow Y$, 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 $\pi_Y^N(t_1) = \pi_Y^N(t_2)$. \square

The next example illustrates Definition 17 using Example 2.

Example 18. Consider again the nested attribute

HALFTONING(Brightness, INPUT[Level], OUTPUT[Bit])

from Example 2. Recall from this example that the designer may want to specify the constraint that the length of the input sequence determines the length of the output sequence. This functional dependency formally reads as

HALFTONING(λ , INPUT[λ], λ) \rightarrow HALFTONING(λ , λ , OUTPUT[λ]).

An example of an FD that will, in general, be violated is:

HALFTONING(Brightness, INPUT[λ], λ) \rightarrow HALFTONING(λ , INPUT[Level], λ).

For instance, the two tuples

$$\left(\frac{3}{2}, \left[0, 0, \frac{1}{2}, 1\right], [0, 0, 0, 1]\right) \quad \text{and} \quad \left(\frac{3}{2}, \left[1, 0, 0, \frac{1}{2}\right], [1, 0, 0, 1]\right)$$

have the same projection on $\text{HALFTONING}(\text{Brightness}, \text{INPUT}[\lambda], \lambda)$ but different projections on $\text{HALFTONING}(\lambda, \text{INPUT}[\text{Level}], \lambda)$. \square

We will now formalise the notion of a multivalued dependency in the presence of record and list constructor.

Definition 19. Let $N \in \mathcal{N}_{\{\text{record}, \text{list}\}}$ be a nested attribute. A *multivalued dependency* on N is an expression of the form $X \twoheadrightarrow Y$ where $X, Y \in \text{Sub}(N)$. A set $r \subseteq \text{dom}(N)$ satisfies the multivalued dependency $X \twoheadrightarrow Y$ on N if and only if for all $t_1, t_2 \in r$ with $\pi_X^N(t_1) = \pi_X^N(t_2)$ there is some $t \in r$ with $\pi_{X \sqcup Y}^N(t) = \pi_{X \sqcup Y}^N(t_1)$ and $\pi_{X \sqcup Y^c}^N(t) = \pi_{X \sqcup Y^c}^N(t_2)$. \square

Intuitively, an instance r exhibits the MVD $X \twoheadrightarrow Y$ whenever every 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 20. The second constraint on

$$\text{HALFTONING}(\text{Brightness}, \text{INPUT}[\text{Level}], \text{OUTPUT}[\text{Bit}])$$

from [Example 2](#) now reads formally as:

$$\text{HALFTONING}(\text{Brightness}, \text{INPUT}[\lambda], \lambda) \twoheadrightarrow \text{HALFTONING}(\lambda, \text{INPUT}[\text{Level}], \lambda).$$

This MVD says that each projection on $\text{HALFTONING}(\text{Brightness}, \text{INPUT}[\lambda], \lambda)$ determines the set of projections on $\text{HALFTONING}(\lambda, \text{INPUT}[\text{Level}], \lambda)$ independently from the projections on $\text{HALFTONING}(\lambda, \lambda, \text{OUTPUT}[\text{Bit}])$. \square

[Definition 19](#) also covers MVDs from the relational model of data. For example, the MVD $\text{Title} \twoheadrightarrow \text{Actor}$ on the relation schema $\text{DVD} = \{\text{Title}, \text{Actor}, \text{Feature}\}$ becomes $\text{DVD}(\text{Title}, \lambda, \lambda) \twoheadrightarrow \text{DVD}(\lambda, \text{Actor}, \lambda)$ on the nested attribute $\text{DVD}(\text{Title}, \text{Actor}, \text{Feature})$.

Fagin proves in [18] 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 $N \in \mathcal{N}$ and $X, Y \in \text{Sub}(N)$. Let $r_1 \subseteq \text{dom}(X)$ and $r_2 \subseteq \text{dom}(Y)$. Then $r_1 \bowtie r_2 = \{t \in \text{dom}(X \sqcup Y) \mid \exists t_1 \in r_1, t_2 \in r_2. \pi_X^{X \sqcup Y}(t) = t_1 \text{ and } \pi_Y^{X \sqcup Y}(t) = t_2\}$ is called the *generalised natural join* $r_1 \bowtie r_2$ of r_1 and r_2 . The *projection* $\pi_X(r)$ of $r \subseteq \text{dom}(N)$ on $X \in \text{Sub}(N)$ is defined as $\{\pi_X^N(t) \mid t \in r\}$.

Theorem 21 ([22, Theorem 4.1]). Let $N \in \mathcal{N}_{\{\text{record}, \text{list}\}}$, $r \subseteq \text{dom}(N)$ and $X \twoheadrightarrow Y$ a multivalued dependency on N . Then $X \twoheadrightarrow Y$ is satisfied by r if and only if $r = \pi_{X \sqcup Y}(r) \bowtie \pi_{X \sqcup Y^c}(r)$. \square

4.2. FDs in the general case

While [Definition 17](#) suffices for FDs in the presence of records and lists it is not expressive enough in the presence of set or multiset constructor. The following definition is more general and meets the additional requirements.

Definition 22. Let $N \in \mathcal{N}$ be a nested attribute. A *functional dependency* on N is an expression of the form $\mathcal{X} \rightarrow \mathcal{Y}$ where $\mathcal{X}, \mathcal{Y} \subseteq \text{Sub}(N)$ are non-empty. A set $r \subseteq \text{dom}(N)$ satisfies the FD $\mathcal{X} \rightarrow \mathcal{Y}$ on N , denoted by $\models_r \mathcal{X} \rightarrow \mathcal{Y}$, if and only if for all $t_1, t_2 \in r$ the following holds: if $\pi_X^N(t_1) = \pi_X^N(t_2)$ for all $X \in \mathcal{X}$, then $\pi_Y^N(t_1) = \pi_Y^N(t_2)$ for all $Y \in \mathcal{Y}$. \square

The next example formalises the functional dependency from [Example 3](#), and illustrates the proper gain of expressiveness achieved by [Definition 22](#) in the presence of set and multiset constructor.

Example 23. Consider again the nested attribute

$$\text{PROFILE}(\text{Customer}, \text{BAG}\langle \text{ITEM}(\text{Article}, \text{Price}) \rangle, \text{Discount}).$$

Intuitively, the customers who bought the same bag of items should receive the same discount. Formally, this constraint can be specified as the functional dependency

$$\text{PROFILE}(\lambda, \text{BAG}(\text{ITEM}(\text{Article}, \text{Price})), \lambda) \rightarrow \text{PROFILE}(\lambda, \lambda, \text{Discount}).$$

Note that this FD is different from the FD

$$\{\text{PROFILE}(\lambda, \text{BAG}(\text{ITEM}(\text{Article}, \lambda)), \lambda), \text{PROFILE}(\lambda, \text{BAG}(\text{ITEM}(\lambda, \text{Price})), \lambda)\} \rightarrow \text{PROFILE}(\lambda, \lambda, \text{Discount})$$

as demonstrated by the two nested data elements

$$t_1 = (\text{Homer}, \langle (\text{Chocolate}, 3\$), (\text{Chocolate}, 3\$), (\text{Beer}, 4\$), (\text{Beer}, 5\$), 2\$ \rangle)$$

and

$$t_2 = (\text{Bart}, \langle (\text{Chocolate}, 4\$), (\text{Chocolate}, 5\$), (\text{Beer}, 3\$), (\text{Beer}, 3\$), 0\$ \rangle).$$

In fact, $\{t_1, t_2\}$ satisfy the first FD, but do not satisfy the second FD. \square

For $\mathcal{X} \subseteq \text{Sub}(N)$ let $\vartheta(\mathcal{X})$ contain all the extended join-irreducibles of N that are subattributes of some element of \mathcal{X} and which are \leq -maximal with this property, i.e.,

$$\vartheta(\mathcal{X}) = \max_{\leq} \{Y \in \mathcal{E}(N) \mid Y \leq X \text{ for some } X \in \mathcal{X}\}.$$

In case that \mathcal{X} is the singleton $\{X\}$ we write $\vartheta(X)$. It is not difficult to see that a database instance $r \subseteq \text{dom}(N)$ satisfies the FD $\mathcal{X} \rightarrow \mathcal{Y}$ if and only if r satisfies the FD $\vartheta(\mathcal{X}) \rightarrow \vartheta(\mathcal{Y})$. We may therefore assume without loss of generality that every FD $\mathcal{X} \rightarrow \mathcal{Y}$ is of the form $\mathcal{X} = \vartheta(\mathcal{X})$ and $\mathcal{Y} = \vartheta(\mathcal{Y})$. Moreover, every FD $X \rightarrow Y$ on $N \in \mathcal{N}_{\{\text{record}, \text{list}\}}$ is satisfied by precisely the same database instances that satisfy $\vartheta(X) \rightarrow \vartheta(Y)$. In the same way we can think of MVDs $X \twoheadrightarrow Y$ on $N \in \mathcal{N}_{\{\text{record}, \text{list}\}}$ as expressions $\vartheta(X) \twoheadrightarrow \vartheta(Y)$.

4.3. Boolean dependencies in the general case

A functional dependency can be viewed as a special case of what we will introduce now as a Boolean dependency on a nested attribute. This class also generalises the class of Boolean dependencies on the relation schemata [30].

Definition 24. Let $N \in \mathcal{N}$. The set of *Boolean dependencies* on N is the smallest set $Bd(N)$ with the following properties:

- (1) $\mathcal{E}(N) \subseteq Bd(N)$,
- (2) if $X \in Bd(N)$, then $\neg X \in Bd(N)$,
- (3) if $X, Y \in Bd(N)$, then $(X \wedge Y), (X \vee Y), (X \Rightarrow Y) \in Bd(N)$. \square

Before formally defining the satisfaction of Boolean dependencies we would like to point out a subtle difference to the definition of satisfaction for functional dependencies. Since a general Boolean dependency may use the Boolean connective \neg for negation, a straightforward definition of satisfaction is not meaningful for the Boolean dependencies with negation. For example, consider the nested attribute $\text{COURSE}(\text{No}, \text{Name}, \text{TEAM}\{\text{Teacher}\})$ together with the Boolean dependency

$$\text{COURSE}(\lambda, \lambda, \text{TEAM}\{\text{Teacher}\}) \Rightarrow \neg \text{COURSE}(\text{No}, \lambda, \lambda)$$

that expresses the fact that two *distinct* course offerings with the same team of teachers must have different course numbers. Such a dependency may be specified if each tuple represents information on a different course. Applying a straightforward extension of the definition of satisfaction it is impossible to express the intended meaning by the Boolean dependency above.

Definition 25. Let $N \in \mathcal{N}$, $\varphi \in Bd(N)$ and $t_1, t_2 \in \text{dom}(N)$ two distinct nested data elements. We say that $\{t_1, t_2\}$ *satisfies* the Boolean dependency φ , denoted by $\models_{\{t_1, t_2\}} \varphi$, if and only if the following holds:

- (1) if $\varphi = X \in \mathcal{E}(N)$, then $\models_{\{t_1, t_2\}} \varphi$ if and only if $\pi_X^N(t_1) = \pi_X^N(t_2)$,

- (2) if $\varphi = \neg\sigma$ for $\sigma \in Bd(N)$, then $\models_{\{t_1, t_2\}} \varphi$ if and only if not $\models_{\{t_1, t_2\}} \sigma$,
(3) if $\varphi = (\varphi_1 \wedge \varphi_2)$ for $\varphi_1, \varphi_2 \in Bd(N)$, then $\models_{\{t_1, t_2\}} \varphi$ if and only if $\models_{\{t_1, t_2\}} \varphi_1$ and $\models_{\{t_1, t_2\}} \varphi_2$,
(4) if $\varphi = (\varphi_1 \vee \varphi_2)$ for $\varphi_1, \varphi_2 \in Bd(N)$, then $\models_{\{t_1, t_2\}} \varphi$ if and only if $\models_{\{t_1, t_2\}} \varphi_1$ or $\models_{\{t_1, t_2\}} \varphi_2$,
(5) if $\varphi = (\varphi_1 \Rightarrow \varphi_2)$ for $\varphi_1, \varphi_2 \in Bd(N)$, then $\models_{\{t_1, t_2\}} \varphi$ if and only if $\models_{\{t_1, t_2\}} \varphi_2$ whenever $\models_{\{t_1, t_2\}} \varphi_1$.

More generally, we say that $r \subseteq dom(N)$ satisfies the Boolean dependency φ , denoted by $\models_r \varphi$ if and only if for all $t_1, t_2 \in r$ it is the case that if $t_1 \neq t_2$, then $\models_{\{t_1, t_2\}} \varphi$. \square

Since database instances are sets and therefore do not permit duplicates there is a Boolean dependency on N that is satisfied by every $r \subseteq dom(N)$. In fact, this is the Boolean dependency stating that any two distinct elements $t_1, t_2 \in r$ must have different projections on at least one \leq -maximal extended join-irreducible element of N . Formally, this Boolean dependency is

$$\phi_N = \neg X_1 \vee \dots \vee \neg X_k$$

where X_1, \dots, X_k denote all the \leq -maximal extended join-irreducibles of N .

4.4. The implication of data dependencies

Let \mathcal{C} denote a class of dependencies, $\Sigma \cup \{\varphi\}$ be a set of dependencies from \mathcal{C} all defined on the same nested attribute N . We say that Σ (finitely) implies φ , denoted by $\Sigma \models \varphi$ ($\Sigma \models_{\text{fin}} \varphi$) if and only if all (finite) $r \subseteq dom(N)$ that satisfy all dependencies in Σ also satisfy φ . Furthermore, Σ implies φ 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 φ . In this paper we deal with classes \mathcal{C} of data dependencies for which finite and unrestricted implication coincide [21,22]. Note that for the class of Boolean dependencies this is immediate from Definition 25. As it will turn out in this paper (finite) implication for \mathcal{C} even coincides with implication in the world of two-element instances. The implication problem for a class \mathcal{C} of dependencies is to decide whether for an arbitrary nested attribute N and an arbitrary set $\Sigma \cup \{\varphi\}$ of dependencies on N from \mathcal{C} the following holds: $\Sigma \models \varphi$.

5. Equivalence results

In this section we will define the mapping between data dependencies and propositional formulae, and present the equivalence results. We start with some examples that illustrate the techniques that will be used in the proof arguments.

We assume familiarity with basic notions from the classical Boolean propositional logic [16]. We will be particularly interested in propositional formulae of the form

$$(U_1 \wedge \dots \wedge U_n) \Rightarrow (V_1 \wedge \dots \wedge V_m) \quad \text{or} \quad (U_1 \wedge \dots \wedge U_n) \Rightarrow ((V_1 \wedge \dots \wedge V_m) \vee (W_1 \wedge \dots \wedge W_k))$$

where U_i, V_j, W_l denote propositional variables. We assume that the conjunction of 0 propositional variables is *true*, that negation \neg binds stronger than any other Boolean connectives, and that conjunction \wedge and disjunction \vee bind stronger than implication \Rightarrow . The formulae above become $U_1 \wedge \dots \wedge U_n \Rightarrow V_1 \wedge \dots \wedge V_m$ and $U_1 \wedge \dots \wedge U_n \Rightarrow (V_1 \wedge \dots \wedge V_m) \vee (W_1 \wedge \dots \wedge W_k)$, respectively. The satisfaction of a propositional formula φ' by a truth assignment θ is denoted by $\models_{\theta} \varphi'$.

5.1. An example for the case of lists

Consider the nested attribute

$$\text{HALFTONING}(\text{Brightness}, \text{INPUT}[\text{Level}], \text{OUTPUT}[\text{Bit}])$$

together with the set Σ of FDs and MVDs specified before, i.e.,

$$\begin{aligned} \text{HALFTONING}(\text{Brightness}, \text{INPUT}[\lambda], \lambda) &\rightarrow \text{HALFTONING}(\lambda, \text{INPUT}[\text{Level}], \lambda), \\ \text{HALFTONING}(\lambda, \text{INPUT}[\lambda], \lambda) &\rightarrow \text{HALFTONING}(\lambda, \lambda, \text{OUTPUT}[\lambda]), \quad \text{and} \\ \text{HALFTONING}(\lambda, \lambda, \text{OUTPUT}[\lambda]) &\rightarrow \text{HALFTONING}(\lambda, \text{INPUT}[\lambda], \lambda). \end{aligned}$$

One may ask whether the FD φ

$$\text{HALFTONING}(\text{Brightness}, \text{INPUT}[\text{Level}], \lambda) \rightarrow \text{HALFTONING}(\lambda, \lambda, \text{OUTPUT}[\text{Bit}])$$

is implied by Σ . Consider for instance the two nested data elements

$$\left(\frac{3}{2}, \left[\frac{1}{2}, 0, 1, 0 \right], [1, 0, 1, 0] \right) \quad \text{and} \quad \left(\frac{3}{2}, \left[\frac{1}{2}, 0, 1, 0 \right], [0, 1, 1, 0] \right).$$

Both elements agree on $\text{HALFTONING}(\text{Brightness}, \text{INPUT}[\text{Level}], \text{OUTPUT}[\lambda])$, but have different projections on $\text{HALFTONING}(\lambda, \lambda, \text{OUTPUT}[\text{Bit}])$. Consequently, this two-element instance satisfies all dependencies in Σ , but it does not satisfy φ . Therefore, Σ does not imply φ .

We consider this particular implication problem from a logical point of view. Therefore, we assign propositional variables to the join-irreducibles of $\text{HALFTONING}(\text{Brightness}, \text{INPUT}[\text{Level}], \text{OUTPUT}[\text{Bit}])$ as follows:

- $\text{HALFTONING}(\text{Brightness}, \lambda, \lambda)$ is assigned V_1 ,
- $\text{HALFTONING}(\lambda, \text{INPUT}[\lambda], \lambda)$ is assigned V_2 ,
- $\text{HALFTONING}(\lambda, \text{INPUT}[\text{Level}], \lambda)$ is assigned V_3 ,
- $\text{HALFTONING}(\lambda, \lambda, \text{OUTPUT}[\lambda])$ is assigned V_4 , and
- $\text{HALFTONING}(\lambda, \lambda, \text{OUTPUT}[\text{Bit}])$ is assigned V_5 .

According to the subattribute order \leq between the join-irreducibles the truth assignments to V_1, \dots, V_5 cannot be independent from one another. In this example the structure of the join-irreducibles is encoded by the Horn clauses $V_3 \Rightarrow V_2$ and $V_5 \Rightarrow V_4$. While the correspondence between dependencies and propositional formulae will be defined later on in detail the MVD and FDs in Σ result in the set

$$\Sigma' = \{V_1 \wedge V_2 \Rightarrow V_3 \vee V_5, V_2 \Rightarrow V_4, V_4 \Rightarrow V_5\}$$

of propositional formulae. Moreover, φ is mapped to the propositional formula $\varphi' = V_1 \wedge V_3 \Rightarrow V_5$.

The truth assignment θ with $\theta(V_i) = \text{true}$ if and only if $i \in \{1, 2, 3, 4\}$ satisfies all formulae in Σ' , but violates φ' . That is, φ' is not logically implied by Σ' . Note that the two nested data elements above coincide on projections to precisely those join-irreducibles whose corresponding propositional variable is interpreted as *true* by θ .

5.2. An example for FDs in the general case

Consider the nested attribute N

$$\text{DANCE}(\text{Time}, \text{PARTAKER}\{\text{Name}\}, \text{DUO}\{\text{PAIR}(\text{Girl}, \text{Boy})\}, \text{Rating})$$

together with the following set Σ of FDs

- $\text{DANCE}(\text{Time}, \lambda, \lambda, \lambda) \rightarrow \text{DANCE}(\lambda, \text{PARTAKER}\{\text{Name}\}, \text{DUO}\{\text{PAIR}(\text{Girl}, \text{Boy})\}, \text{Rating})$,
- $\text{DANCE}(\lambda, \text{PARTAKER}\{\text{Name}\}, \lambda, \lambda) \rightarrow \{\text{DANCE}(\lambda, \lambda, \text{DUO}\{\text{PAIR}(\text{Girl}, \lambda)\}, \lambda), \text{DANCE}(\lambda, \lambda, \text{DUO}\{\text{PAIR}(\lambda, \text{Boy})\}, \lambda)\}$,
- $\{\text{DANCE}(\lambda, \lambda, \text{DUO}\{\text{PAIR}(\text{Girl}, \lambda)\}, \lambda), \text{DANCE}(\lambda, \lambda, \text{DUO}\{\text{PAIR}(\lambda, \text{Boy})\}, \lambda)\} \rightarrow \text{DANCE}(\lambda, \text{PARTAKER}\{\text{Name}\}, \lambda, \lambda)$, and
- $\text{DANCE}(\lambda, \lambda, \text{DUO}\{\text{PAIR}(\text{Girl}, \text{Boy})\}, \lambda) \rightarrow \text{DANCE}(\lambda, \lambda, \lambda, \text{Rating})$.

The attribute N models dancing classes in which partakers are divided into pairs each consisting of a girl and a boy. The first FD says informally that the time of the course determines everything else, i.e., $\text{DANCE}(\text{Time}, \lambda, \lambda, \lambda)$ is a key. The second FD says informally that the set of partakers determines the set of boys and the set of girls. Vice versa, the third FD says informally that the set of girls and the set of boys together determine the set of partakers. Finally, the last FD says informally that the set of pairs determines the rating. That is, the same combination of boys and girls results in the same rating. Suppose we want to decide if the single FD φ

$$\text{DANCE}(\lambda, \text{PARTAKER}\{\text{Name}\}, \lambda, \lambda) \rightarrow \text{DANCE}(\lambda, \lambda, \lambda, \text{Rating})$$

is a consequence of Σ . The nested two-tuple relation r consisting of

$$t_1 = (29.2.1600, \{\text{Dulcinea, Theresa, Don Quixote, Sancho}\}, \{(\text{Dulcinea, Don Quixote}), (\text{Theresa, Sancho})\}, 10) \quad \text{and}$$

$$t_2 = (1.3.1600, \{\text{Dulcinea, Theresa, Don Quixote, Sancho}\}, \{(\text{Dulcinea, Sancho}), (\text{Theresa, Don Quixote})\}, 3)$$

satisfies all FDs in Σ , but violates φ . Therefore, we have found a counterexample relation r for the implication of φ by Σ . We consider the problem now from a logical point of view. First, the extended join-irreducibles of N are mapped to propositional variables as follows:

- DANCE(Time, $\lambda, \lambda, \lambda$) is V_1 ,
- DANCE(λ , PARTAKER{Name}, λ, λ) is V_2 ,
- DANCE(λ , PARTAKER{ λ }, λ, λ) is V_3 ,
- DANCE(λ, λ , DUO{PAIR(Girl, Boy)}, λ) is V_4 ,
- DANCE(λ, λ , DUO{PAIR(Girl, λ)}, λ) is V_5 ,
- DANCE(λ, λ , DUO{PAIR(λ , Boy)}, λ) is V_6 ,
- DANCE(λ, λ , DUO{PAIR(λ, λ)}, λ) is V_7 , and
- DANCE($\lambda, \lambda, \lambda$, Rating) is V_8 .

The set Σ'_N of Horn clauses that encodes the structure of the database schema N is then given by

$$V_2 \Rightarrow V_3, \quad V_4 \Rightarrow V_5, \quad V_4 \Rightarrow V_6, \quad V_5 \Rightarrow V_7, \quad V_6 \Rightarrow V_7.$$

The set Σ of FDs has the following corresponding set Σ' of Horn clauses

- $V_1 \Rightarrow V_2, V_1 \Rightarrow V_4, V_1 \Rightarrow V_8$,
- $V_2 \Rightarrow V_5, V_2 \Rightarrow V_6$,
- $V_5 \wedge V_6 \Rightarrow V_2$, and
- $V_4 \Rightarrow V_8$

and φ corresponds to the single Horn clause $\varphi' = V_2 \Rightarrow V_8$. The truth assignment θ with $\theta(V_i) = \text{true}$ iff $i \in \{2, 3, 5, 6, 7\}$ satisfies all clauses in $\Sigma' \cup \Sigma'_N$, but violates φ' . Therefore, φ' is not a logical consequence of $\Sigma' \cup \Sigma'_N$. Most importantly, note the correspondence between the nested counterexample relation r and the truth assignment θ . In fact, the tuples t_1 and t_2 agree exactly on those extended join-irreducibles whose corresponding propositional variable is assigned the truth value *true* by θ . This turns out to be the decisive argument for showing the equivalence.

5.3. Mapping data dependencies to propositional formulae

Let $\phi : \mathcal{E}(N) \rightarrow \mathcal{V}$ denote a bijection between the extended join-irreducibles of N and the set \mathcal{V} of Boolean propositional variables. We will now extend this bijection to classes of data dependencies over the nested attribute N and (fragments of) the Boolean propositional formulae over \mathcal{V} .

First, consider the FD $\varphi: \mathcal{X} \rightarrow \mathcal{Y}$ on N where $\vartheta(\mathcal{X}) = \{X_1, \dots, X_n\}$ and $\vartheta(\mathcal{Y}) = \{Y_1, \dots, Y_k\}$. Define $\Phi(\varphi)$ to be the propositional formula

$$\varphi' = \phi(X_1) \wedge \dots \wedge \phi(X_n) \Rightarrow \phi(Y_1) \wedge \dots \wedge \phi(Y_k).$$

Secondly, consider the MVD $\varphi: X \twoheadrightarrow Y$ on N where $\vartheta(X) = \{X_1, \dots, X_n\}$, $\vartheta(Y) = \{Y_1, \dots, Y_k\}$ and $\vartheta(Y_N^C \div X) = \{Z_1, \dots, Z_m\}$. In this case, define $\Phi(\varphi)$ to be the Boolean propositional formula

$$\varphi' = \phi(X_1) \wedge \dots \wedge \phi(X_n) \Rightarrow (\phi(Y_1) \wedge \dots \wedge \phi(Y_k)) \vee (\phi(Z_1) \wedge \dots \wedge \phi(Z_m)).$$

The MVD

$$\text{HALFTONING}(\text{Brightness}, \text{INPUT}[\lambda], \lambda) \twoheadrightarrow \text{HALFTONING}(\lambda, \text{INPUT}[\text{Level}], \lambda),$$

for example, results in $V_1 \wedge V_2 \Rightarrow V_3 \vee V_5$. Finally, we recursively define the mapping of Boolean dependencies φ to their equivalent Boolean propositional formulae $\Phi(\varphi) = \varphi'$. If $\varphi = X$ is an extended join-irreducible of N , then let $\varphi' = \phi(X)$. The rest of the mapping is straightforward:

- for $\varphi = \neg\sigma$ we have $\varphi' = \neg\sigma'$,
- for $\varphi = (\varphi_1 \vee \varphi_2)$ we have $\varphi' = (\varphi'_1 \vee \varphi'_2)$,
- for $\varphi = (\varphi_1 \wedge \varphi_2)$ we have $\varphi' = (\varphi'_1 \wedge \varphi'_2)$, and
- for $\varphi = (\varphi_1 \Rightarrow \varphi_2)$ we have $\varphi' = (\varphi'_1 \Rightarrow \varphi'_2)$.

If Σ is a set of dependencies on N , then let $\Sigma' = \{\sigma' \mid \sigma \in \Sigma\}$ denote the corresponding set of propositional formulae over \mathcal{V} . Furthermore, the set

$$\Sigma'_N = \{\phi(U) \Rightarrow \phi(V) \mid U, V \in \mathcal{E}(N), U \text{ covers}^2 V\}$$

denotes those formulae which encode the structure of (the extended join-irreducibles of) N .

5.4. The main results

We will now present the main results of this paper. They generalise results from the relational data model [17,30,31] where

- (1) only nested attributes are considered that result from a single application of the record constructor to a finite number of flat attributes,
- (2) the join-irreducibles of these nested attributes form an anti-chain, and
- (3) it is sufficient to consider join-irreducibles only.

The first main result captures two cases. In the first case $\Sigma \cup \{\varphi\}$ is a set of FDs and MVDs on a nested attribute $N \in \mathcal{N}_{\{\text{record}, \text{list}\}}$. That is, we allow arbitrary finite applications of record and list constructor. While it is still sufficient to consider the join-irreducibles of N , they no longer form an anti-chain with respect to \leq . In the second case $\Sigma \cup \{\varphi\}$ is a set of FDs on the nested attribute $N \in \mathcal{N}$. That is, we allow arbitrary finite applications of record, list, set and multiset constructor. In order to obtain an equivalence to the propositional Horn clauses we need to consider extended join-irreducibles of N . Again, these extended join-irreducibles do not form an anti-chain.

If $\Sigma \cup \{\varphi\}$ is a set of FDs and MVDs on the nested attribute N , then we implicitly assume that $N \in \mathcal{N}_{\{\text{record}, \text{list}\}}$.

Theorem 26 (Equivalence Theorem for FDs and MVDs). *Let N be a nested attribute, and $\Sigma \cup \{\varphi\}$ either a set of FDs and MVDs on N or a set of FDs on N . Let Σ'_N denote the propositional formulae which encode the structure of N , and Σ' denote the corresponding set of propositional formulae for Σ . Then*

- (1) Σ implies φ ,
- (2) Σ implies φ in the world of two-element instances, and
- (3) $\Sigma' \cup \Sigma'_N$ logically implies φ'

are equivalent. \square

The second main result generalises the second case from the previous theorem to the class of Boolean dependencies. In order to capture the implication of these Boolean dependencies we require the propositional formula ϕ_N . This is already the case in the relational model of data [31].

Theorem 27 (Equivalence Theorem for Boolean Dependencies). *Let $N \in \mathcal{N}$ be a nested attribute, and $\Sigma \cup \{\varphi\}$ a set of Boolean dependencies on N . Let Σ'_N denote the propositional formulae which encode the structure of N , and Σ' denote the corresponding set of propositional formulae for Σ . Then*

- (1) Σ implies φ ,
- (2) Σ implies φ in the world of two-element instances, and
- (3) $\Sigma' \cup \Sigma'_N \cup \{\phi_N\}$ logically implies φ'

are equivalent. \square

² U covers V iff $U < V$ and for all $W \in \mathcal{E}(N)$ 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 [3].

Using [Example 3](#) the following illustration shows the need of the formula ϕ_N to achieve the equivalence between the implication of Boolean dependencies and Boolean propositional formulae.

Example 28. Consider again the nested attribute N

$$\text{PROFILE}(\text{Customer}, \text{BAG}\langle \text{ITEM}(\text{Article}, \text{Price}) \rangle, \text{Discount})$$

together with the functional dependency σ

$$\text{PROFILE}(\lambda, \text{BAG}\langle \text{ITEM}(\text{Article}, \text{Price}) \rangle, \lambda) \rightarrow \text{PROFILE}(\lambda, \lambda, \text{Discount}).$$

This functional dependency implies that the Boolean dependency φ

$$\neg \text{PROFILE}(\text{Customer}, \lambda, \lambda) \vee \neg \text{PROFILE}(\lambda, \text{BAG}\langle \text{ITEM}(\text{Article}, \text{Price}) \rangle, \lambda).$$

Indeed, if two distinct elements agree on $\text{PROFILE}(\text{Customer}, \lambda, \lambda)$, then they must have different projections on $\text{PROFILE}(\lambda, \text{BAG}\langle \text{ITEM}(\text{Article}, \text{Price}) \rangle, \lambda)$ since the elements would not be distinct otherwise. Vice versa, if two distinct elements agree on $\text{PROFILE}(\lambda, \text{BAG}\langle \text{ITEM}(\text{Article}, \text{Price}) \rangle, \lambda)$ (and therefore also on $\text{PROFILE}(\lambda, \lambda, \text{Discount})$), their projections on $\text{PROFILE}(\text{Customer}, \lambda, \lambda)$ must be different.

Consider the implication from a logical point of view. Extended join-irreducibles are mapped to propositional variables as follows:

- $\text{PROFILE}(\text{Customer}, \lambda, \lambda)$ is mapped to V_1 ,
- $\text{PROFILE}(\lambda, \text{BAG}\langle \text{ITEM}(\lambda, \lambda) \rangle, \lambda)$ is mapped to V_2 ,
- $\text{PROFILE}(\lambda, \text{BAG}\langle \text{ITEM}(\text{Article}, \lambda) \rangle, \lambda)$ is mapped to V_3 ,
- $\text{PROFILE}(\lambda, \text{BAG}\langle \text{ITEM}(\lambda, \text{Price}) \rangle, \lambda)$ is mapped to V_4 ,
- $\text{PROFILE}(\lambda, \text{BAG}\langle \text{ITEM}(\text{Article}, \text{Price}) \rangle, \lambda)$ is mapped to V_5 , and
- $\text{PROFILE}(\lambda, \lambda, \text{Discount})$ is mapped to V_6 .

The FD σ reads translates into the Horn clause $\sigma' = V_5 \Rightarrow V_6$, and the Boolean dependency φ into $\varphi' = \neg V_1 \vee \neg V_5$. Moreover,

$$\Sigma'_N = \{V_5 \Rightarrow V_3, V_5 \Rightarrow V_4, V_3 \Rightarrow V_2, V_4 \Rightarrow V_2\}.$$

The truth assignment θ that assigns *true* to V_i for all i with $1 \leq i \leq 6$ shows that φ' is not implied by $\Sigma'_N \cup \{\sigma'\}$. It illustrates the necessity of having

$$\phi_N = \neg V_1 \vee \neg V_5 \vee \neg V_6$$

among the premises. Indeed, $\Sigma'_N \cup \{\sigma', \phi_N\}$ implies φ' . \square

Remark 29. Similar to the relational data model [31] one may allow multisets $r \subseteq \text{dom}(N)$ as possible database instances. In this case the propositional formula ϕ_N is not required in [Theorem 27](#), and the Boolean dependency φ is implied by Σ if and only if its corresponding propositional formula φ' is logically implied by $\Sigma' \cup \Sigma'_N$. \square

5.5. Proofs

We will now verify [Theorems 26](#) and [27](#). Therefore, we will give a complete proof of [Theorem 27](#) which will also cover the case of [Theorem 26](#) in which the set $\Sigma \cup \{\varphi\}$ contains only FDs. It then remains to prove the result on MVDs as well which are not Boolean dependencies.

5.5.1. The Boolean dependencies

We start off by showing the equivalence of (1) and (2) in [Theorem 27](#). It is immediate that (1) implies (2) since every two-element instance is also an instance. The converse implication follows from [Definition 25](#). Assume that (1) does not hold. That is, there is some $r \subseteq \text{dom}(N)$ such that $\models_r \sigma$ for all $\sigma \in \Sigma$, but not $\models_r \varphi$. According to [Definition 25](#) there must be some $t_1, t_2 \in r$ such that $t_1 \neq t_2$ and not $\models_{\{t_1, t_2\}} \varphi$. Since $\{t_1, t_2\} \subseteq r$ we must have $\models_{\{t_1, t_2\}} \sigma$ for all $\sigma \in \Sigma$. Consequently, Σ does not imply φ in the world of two-element instances, i.e., not (2).

In order to complete the proof of [Theorem 27](#) it remains to show the equivalence between (2) and (3). The key idea is to define truth assignments based on the two-element instances and vice versa. In fact, one interprets a variable as *true* precisely if the two nested data elements coincide on their projections to the corresponding extended join-irreducible of that variable.

Lemma 30. *Let φ be a Boolean dependency on the nested attribute N , and $r = \{t_1, t_2\} \subseteq \text{dom}(N)$ such that $t_1 \neq t_2$. Then $\models_r \varphi$ if and only if $\models_{\theta_r} \varphi'$ where*

$$\theta_r(V) = \begin{cases} \text{true} & , \quad \text{if } \pi_{\phi^{-1}(V)}^N(t_1) = \pi_{\phi^{-1}(V)}^N(t_2) \\ \text{false} & , \quad \text{else} \end{cases}$$

for all $V \in \phi(\mathcal{E}(N))$.

Proof. The proof is done by induction on the structure of φ . Let $\varphi = X \in \mathcal{E}(N)$. We then have $\models_{\{t_1, t_2\}} X$ if and only if $\pi_X^N(t_1) = \pi_X^N(t_2)$ by [Definition 25](#). The last condition, however, is equivalent to $\theta_r(\phi(X)) = \text{true}$. This shows the start of the induction. The induction steps are a straightforward application of [Definition 25](#). \square

We are now prepared to show the equivalence between (2) and (3). Suppose (2) does not hold. Then there are $t_1, t_2 \in \text{dom}(N)$ such that $t_1 \neq t_2$ and $\models_{\{t_1, t_2\}} \sigma$ for all $\sigma \in \Sigma$, but not $\models_{\{t_1, t_2\}} \varphi$. According to [Lemma 30](#) we know that $\models_{\theta_{\{t_1, t_2\}}} \sigma'$ for all $\sigma' \in \Sigma'$ and not $\models_{\theta_{\{t_1, t_2\}}} \varphi'$. Since t_1 and t_2 are distinct it follows also that $\models_{\theta_{\{t_1, t_2\}}} \phi_N$. It remains to show that $\models_{\theta_{\{t_1, t_2\}}} V \Rightarrow W$ for all $V \Rightarrow W \in \Sigma'_N$. Let $\phi(X) = V$ and $\phi(Y) = W$ for $X, Y \in \mathcal{E}(N)$. This means that $Y \leq X$. Suppose that $\theta_{\{t_1, t_2\}}(V) = \text{true}$. Then $\pi_X^N(t_1) = \pi_X^N(t_2)$ according to the definition of the truth assignment $\theta_{\{t_1, t_2\}}$. Since $Y \leq X$ it follows that $\pi_Y^N(t_1) = \pi_Y^N(t_2)$ holds as well. This means, however, that $\theta_{\{t_1, t_2\}}(W) = \text{true}$, too. Consequently, φ' is not logically implied by $\Sigma' \cup \Sigma'_N \cup \{\phi_N\}$ as witnessed by $\theta_{\{t_1, t_2\}}$. That means (3) does not hold and it remains to show that (2) implies (3).

Suppose (3) does not hold. Then there is some truth assignment θ which makes every formula in $\Sigma' \cup \Sigma'_N \cup \{\phi_N\}$ *true*, but makes φ' *false*. It is now sufficient to find some $r = \{t_1, t_2\} \subseteq \text{dom}(N)$ such that $t_1 \neq t_2$ and $\theta = \theta_r$. In this case, [Lemma 30](#) shows that $\models_r \sigma$ for all $\sigma \in \Sigma$ and not $\models_r \varphi$, i.e., (2) does not hold.

Let $\mathcal{U}(N) = \{U_1, \dots, U_k\}$, and $\mathcal{X}_{U_i}^+ = \{X \in \mathcal{E}(N) \mid X \leq U_i \text{ and } \theta(\phi(X)) = \text{true}\} \cup \{\lambda_N\}$. Note that $\mathcal{X}_{U_i}^+$ is closed downwards with respect to \leq since θ satisfies Σ'_N . Moreover, let $\mathcal{X}^+ = \{X_1 \sqcup \dots \sqcup X_k \mid X_i \in \mathcal{X}_{U_i}^+ \text{ for } 1 \leq i \leq k\}$. In this case, \mathcal{X}^+ is non-empty, closed downwards with respect to \leq , and closed under the join of reconcilable elements. \mathcal{X}^+ is non-empty since $\lambda_N \in \mathcal{X}_{U_i}^+$ for $i = 1, \dots, k$. It is closed downwards with respect to \leq since every $\mathcal{X}_{U_i}^+$ has the same property. In order to see that \mathcal{X}^+ is closed under the join of reconcilable elements suppose that $X = X_1 \sqcup \dots \sqcup X_k, X' = X'_1 \sqcup \dots \sqcup X'_k \in \mathcal{X}^+$ are reconcilable. Since $X \sqcap U_i = X_i$ and $X' \sqcap U_i = X'_i$ for $i = 1, \dots, k$ it must be the case that X_i and X'_i are comparable with respect to \leq for all $i = 1, \dots, k$. This, however, means that $X \sqcup X' \in \mathcal{X}^+$ by definition of \mathcal{X}^+ . [Lemma 11](#) then shows the existence of two elements $t_1, t_2 \in \text{dom}(N)$ with the property that for all $X \in \text{Sub}(N)$ we have $\pi_X^N(t_1) = \pi_X^N(t_2)$ if and only if $X \in \mathcal{X}^+$ holds. Since for all $X \in \mathcal{E}(N)$ we have that $\theta(\phi(X)) = \text{true}$ if and only if $X \in \mathcal{X}^+$ if and only if $\pi_X^N(t_1) = \pi_X^N(t_2)$ holds, it follows that $\theta = \theta_{\{t_1, t_2\}}$. Moreover, since $\theta(\phi_N) = \text{true}$ and $\theta = \theta_{\{t_1, t_2\}}$ the two elements t_1 and t_2 must be distinct as well. This shows that (2) does not hold and the proof of [Theorem 27](#) is complete.

5.5.2. Multivalued dependencies

Since multivalued dependencies are not Boolean dependencies [Theorem 26](#) is not a special case of [Theorem 27](#). Throughout this paragraph we assume that $N \in \mathcal{N}_{\{\text{record}, \text{list}\}}$. The proof of [Theorem 26](#) follows the same ideas as that of [Theorem 27](#), and the main arguments involve an extension of [Lemma 30](#) to MVDs and that the implication of FDs and MVDs over two-element instances implies implication of FDs and MVDs over arbitrary finite instances. Note that finite and unrestricted implication coincide for FDs and MVDs [22].

Let $r = \{t_1, t_2\} \subseteq \text{dom}(N)$ a two-element instance over N . The MVD $X \twoheadrightarrow Y$ on N is said to hold *actively* in r if and only if r satisfies $X \twoheadrightarrow Y$ and $\pi_X^N(t_1) = \pi_X^N(t_2)$. Consequently, there are two ways the MVD $X \twoheadrightarrow Y$ is satisfied by r : either (1) $\pi_X^N(t_1) \neq \pi_X^N(t_2)$ or else (2) $X \twoheadrightarrow Y$ holds actively in r .

Lemma 31. Let $r = \{t_1, t_2\} \subseteq \text{dom}(N)$. The MVD $X \twoheadrightarrow Y$ on N holds actively in r if and only if

- (1) $\pi_X^N(t_1) = \pi_X^N(t_2)$, and
- (2) $\pi_Y^N(t_1) = \pi_Y^N(t_2)$ or $\pi_{Y^C}^N(t_1) = \pi_{Y^C}^N(t_2)$.

Proof. If (1) and (2) hold, then $X \twoheadrightarrow Y$ holds actively in r . Conversely, assume that $X \twoheadrightarrow Y$ holds actively in r . By definition, (1) holds. Consequently, there must be a tuple $t \in r$ with $\pi_{X \sqcup Y}^N(t) = \pi_{X \sqcup Y}^N(t_1)$ and $\pi_{X \sqcup Y^C}^N(t) = \pi_{X \sqcup Y^C}^N(t_2)$. Since $t = t_1$ or $t = t_2$ we have $\pi_Y^N(t_1) = \pi_Y^N(t_2)$ or $\pi_{Y^C}^N(t_1) = \pi_{Y^C}^N(t_2)$. \square

Lemma 32. Let $r, r' \subseteq \text{dom}(N)$ be two-element instances over N . Assume that whenever the two elements of r have the same projection on some join-irreducible M of N , then the two elements of r have the same projection on M . Then each MVD that holds actively in r also holds actively in r' .

Proof. Assume $X \twoheadrightarrow Y$ holds actively in r . For $t_1, t_2 \in r$ we have $\pi_X^N(t_1) = \pi_X^N(t_2)$. Consequently, $\pi_X^N(t'_1) = \pi_X^N(t'_2)$ for $t'_1, t'_2 \in r'$. According to Lemma 31 we have $\pi_Y^N(t_1) = \pi_Y^N(t_2)$ or $\pi_{Y^C}^N(t_1) = \pi_{Y^C}^N(t_2)$. It follows that $\pi_Y^N(t'_1) = \pi_Y^N(t'_2)$ or $\pi_{Y^C}^N(t'_1) = \pi_{Y^C}^N(t'_2)$ holds as well, i.e., $X \twoheadrightarrow Y$ holds actively in r' . \square

Lemma 33. Assume that $r \subseteq \text{dom}(N)$ is some finite instance over N , Σ a set of FDs and MVDs on N , and φ a single FD or MVD on N . Suppose that r satisfies all FDs and MVDs in Σ , but does not satisfy φ . 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 φ .

Proof. Consider first the case where φ is an FD. The FD $X \rightarrow Y$ is satisfied by some $s \subseteq \text{dom}(N)$ if and only if s satisfies all FDs $X \rightarrow M$ where $M \in \vartheta(Y)$. Therefore, we can assume without loss of generality that φ is an FD $X \rightarrow M$ where $M \in \mathcal{J}(N)$. Since r does not satisfy φ , there are two elements $t'_1, t'_2 \in \text{dom}(N)$ such that $\pi_X^N(t'_1) = \pi_X^N(t'_2)$, but $\pi_M^N(t'_1) \neq \pi_M^N(t'_2)$. Consider all two-element subsets of r which do not satisfy $X \rightarrow M$. Of all such subsets, let r' be the one for which the maximal number of MVDs hold actively. That is, if r_1 is another two-element subset of r which does not satisfy φ , and if k is the number of MVDs that hold actively in r_1 , then at least k MVDs hold actively in r' . We will show now that all FDs and MVDs in Σ are satisfied by r' .

All FDs in Σ are satisfied by r' since they are satisfied by r , and hence in every subset of r . Let $U \twoheadrightarrow V$ be an MVD in Σ that is not satisfied by r' . We will derive a contradiction. For $r' = \{t_1, t_2\}$ we have $\pi_U^N(t_1) = \pi_U^N(t_2)$. Consequently, $\pi_V^N(t_1) \neq \pi_V^N(t_2)$ and $\pi_{V^C}^N(t_1) \neq \pi_{V^C}^N(t_2)$ must hold or r' would satisfy $U \twoheadrightarrow V$. By assumption, $X \rightarrow M$ is not satisfied by r' , i.e., $\pi_M^N(t_1) \neq \pi_M^N(t_2)$. Consequently, $M \not\leq X$, and $M \leq V$ or $M \leq V^C$. Suppose $M \leq V^C$. Since r satisfies $U \twoheadrightarrow V$ there is some $t \in r$ with $\pi_{U \sqcup V}^N(t) = \pi_{U \sqcup V}^N(t_1)$ and $\pi_{U \sqcup V^C}^N(t) = \pi_{U \sqcup V^C}^N(t_2)$. Let $r'' = \{t, t_1\} \subseteq r$. For $W \in \mathcal{J}(N)$ with $\pi_W^N(t_1) = \pi_W^N(t_2)$ follows that $\pi_W^N(t) = \pi_W^N(t_1)$ holds as well. In particular, $\pi_X^N(t) = \pi_X^N(t_1)$ since $\pi_X^N(t_1) = \pi_X^N(t_2)$. As $M \leq V^C$, $\pi_V^N(t) = \pi_V^N(t_2)$ and $\pi_M^N(t_1) \neq \pi_M^N(t_2)$ we conclude that $\pi_M^N(t) \neq \pi_M^N(t_1)$. According to Lemma 32 every MVD that holds actively in r' also holds actively in r'' , but unlike r' the MVD $U \twoheadrightarrow V$ holds actively in r'' contradicting the maximality of r' . That is, all FDs and MVDs in Σ are satisfied by the two-element instance r' , but $X \rightarrow Y$ is not.

It remains to consider the case where φ is an MVD, say $X \twoheadrightarrow Y$. We say that a pair of elements t_1, t_2 with $\pi_X^N(t_1) = \pi_X^N(t_2)$, $\pi_Y^N(t_1) \neq \pi_Y^N(t_2)$ and $\pi_{Y^C}^N(t_1) \neq \pi_{Y^C}^N(t_2)$ witness the failure of $X \twoheadrightarrow Y$ in an instance $r \subseteq \text{dom}(N)$ if and only if $t_1, t_2 \in r$ and if $t_3 \in \text{dom}(N)$ with $\pi_{X \sqcup Y}^N(t_3) = \pi_{X \sqcup Y}^N(t_2)$ and $\pi_{X \sqcup Y^C}^N(t_3) = \pi_{X \sqcup Y^C}^N(t_1)$ or $t_4 \in \text{dom}(N)$ with $\pi_{X \sqcup Y}^N(t_4) = \pi_{X \sqcup Y}^N(t_1)$ and $\pi_{X \sqcup Y^C}^N(t_4) = \pi_{X \sqcup Y^C}^N(t_2)$ is not an element of the instance r . Thus an MVD is not satisfied by an instance if and only if there is a pair of elements that witness the failure of that MVD. In particular, since the MVD $X \twoheadrightarrow Y$ is not satisfied by r , let $t_1, t_2 \in r$ witness the failure of $X \twoheadrightarrow Y$. Thus t'_3 or t'_4 is not an element of r , where $\pi_{X \sqcup Y}^N(t'_3) = \pi_{X \sqcup Y}^N(t_2)$, $\pi_{X \sqcup Y^C}^N(t'_3) = \pi_{X \sqcup Y^C}^N(t_1)$ and $\pi_{X \sqcup Y}^N(t'_4) = \pi_{X \sqcup Y}^N(t_1)$, $\pi_{X \sqcup Y^C}^N(t'_4) = \pi_{X \sqcup Y^C}^N(t_2)$. Of all two-element subsets of r that witness the failure of $X \twoheadrightarrow Y$, let r' be the one for which the maximal number of MVDs in Σ hold actively. Now we show that every FD and MVD in Σ is satisfied by r' .

As before, every FD in Σ is satisfied by $r' \subseteq r$. Let $U \twoheadrightarrow V$ be an MVD in Σ that is not satisfied by r' . We shall derive a contradiction. Since $U \twoheadrightarrow V$ is not satisfied by r' we know that $\pi_U^N(t_1) = \pi_U^N(t_2)$. Let $\bar{V} = \bigsqcup\{W \in \mathcal{J}(N) \mid W \leq V \text{ and } \pi_W^N(t_1) \neq \pi_W^N(t_2)\}$ and $\overline{V} = \bigsqcup\{W \in \mathcal{J}(N) \mid W \leq V^C \text{ and } \pi_W^N(t_1) \neq \pi_W^N(t_2)\}$. Since $U \twoheadrightarrow V$ is not satisfied by r' we know that $\bar{V} \neq \lambda_N$ and $\overline{V} \neq \lambda_N$. Let $r_1 = \{t_1, t_3\} \subseteq r$ where $t_3 \in r$ with $\pi_{U \sqcup V}^N(t_3) = \pi_{U \sqcup V}^N(t_2)$ and $\pi_{U \sqcup V^C}^N(t_3) = \pi_{U \sqcup V^C}^N(t_1)$. Furthermore, let $r_2 = \{t_1, t_4\} \subseteq r$ where $t_4 \in r$ with

$\pi_{U \sqcup V}^N(t_4) = \pi_{U \sqcup V}^N(t_1)$ and $\pi_{U \sqcup V^C}^N(t_4) = \pi_{U \sqcup V^C}^N(t_2)$. Both r_1 and r_2 are subsets of r since $U \rightarrow V$ is satisfied by r , and they are both two-element instances since $\pi_V^N(t_1) \neq \pi_V^N(t_2)$ and $\pi_{V^C}^N(t_1) \neq \pi_{V^C}^N(t_2)$. Furthermore, for every $W \in \mathcal{J}(N)$ with $\pi_W^N(t_1) = \pi_W^N(t_2)$ follows $\pi_W^N(t_1) = \pi_W^N(t_3)$ and $\pi_W^N(t_1) = \pi_W^N(t_4)$. Consequently, every MVD that actively holds in r' also holds actively in r_1 and r_2 by Lemma 32. Moreover, $U \rightarrow V$ holds also actively in r_1 and r_2 . If $X \rightarrow Y$ is not satisfied by r_1 or not satisfied by r_2 we have derived a contradiction to the maximality of r' , and hence are done. So suppose that $X \rightarrow Y$ is satisfied by r_1 as well as r_2 . Then $X \rightarrow Y$ holds actively in r_1 and r_2 since $\pi_X^N(t_1) = \pi_X^N(t_2) = \pi_X^N(t_3) = \pi_X^N(t_4)$ holds. It follows from Lemma 31 that $\pi_Y^N(t_1) = \pi_Y^N(t_3)$ or $\pi_{Y^C}^N(t_1) = \pi_{Y^C}^N(t_3)$. In the former case $\bar{V} \leq Y^C$, since \bar{V} is superattribute to all those $W \in \mathcal{J}(N)$ with $\pi_W^N(t_1) \neq \pi_W^N(t_3)$. In the latter case $\bar{V} \leq Y$. Thus we know that $\bar{V} \leq Y^C$ or $\bar{V} \leq Y$. Similarly, it follows from our knowledge about r_2 that $\bar{W} \leq Y$ or $\bar{W} \leq Y^C$.

For all $M \in \mathcal{J}(N)$ we have $\pi_M^N(t_1) \neq \pi_M^N(t_2)$ if and only if $M \leq \bar{V} \sqcup \bar{W}$. Let $M \in \mathcal{J}(N)$ with $M \leq \bar{V} \cap Y \cap Y^C$. Then $\pi_M^N(t_1) = \pi_M^N(t_4)$ since $M \leq \bar{V} \leq V$. Since $\{t_1, t_3\}$ and $\{t_1, t_4\}$ both satisfy $X \rightarrow Y$, they also both satisfy $X \rightarrow Y \cap Y^C$ and it follows that $\pi_{Y \cap Y^C}^N(t_1) = \pi_{Y \cap Y^C}^N(t_3) = \pi_{Y \cap Y^C}^N(t_4)$. Consequently, $\pi_M^N(t_4) = \pi_M^N(t_3)$ since $M \leq Y \cap Y^C$. Moreover, $\pi_M^N(t_3) = \pi_M^N(t_2)$ since $M \leq \bar{V} \leq V$. Thus, $\pi_M^N(t_1) = \pi_M^N(t_2)$ for all $M \in \mathcal{J}(N)$ with $M \leq \bar{V} \cap Y \cap Y^C$. The same is true for all $M \in \mathcal{J}(N)$ with $M \leq \bar{W} \cap Y \cap Y^C$. We conclude that if $\bar{V} \sqcup \bar{W} \leq Y$, then $\pi_{Y^C}^N(t_1) = \pi_{Y^C}^N(t_2)$ contradicting the fact that r' does not satisfy $X \rightarrow Y$. Similarly, if $\bar{V} \sqcup \bar{W} \leq Y^C$, then $\pi_Y^N(t_1) = \pi_Y^N(t_2)$ contradicting the fact that r' does not satisfy $X \rightarrow Y$. So, either $\bar{V} \leq Y$ and $\bar{W} \leq Y^C$ holds, or $\bar{V} \leq Y^C$ and $\bar{W} \leq Y$. We assume without loss of generality that the first one holds. For all $W \leq Y$ we have $\pi_W^N(t_1) \neq \pi_W^N(t_2)$ if and only if $W \leq \bar{V}$, and for all $W \leq Y^C$ we have $\pi_W^N(t_1) \neq \pi_W^N(t_2)$ if and only if $W \leq \bar{W}$. Under these conditions $t_3 = t'_3$ and $t_4 = t'_4$. But this is impossible, since $t_3, t_4 \in r$, whereas t'_3 or t'_4 is not in r . \square

Lemma 34. Let φ be an FD or MVD on the nested attribute $N \in \mathcal{N}_{\{\text{list}, \text{record}\}}$, and $r = \{t_1, t_2\} \subseteq \text{dom}(N)$. Then $\models_r \varphi$ if and only if $\models_{\theta_r} \varphi'$ where

$$\theta_r(V) = \begin{cases} \text{true} & , \quad \text{if } \pi_{\phi^{-1}(V)}^N(t_1) = \pi_{\phi^{-1}(V)}^N(t_2) \\ \text{false} & , \quad \text{else} \end{cases}$$

for all $V \in \phi(\mathcal{J}(N))$.

Proof. Let φ be an FD, say $X \rightarrow Y$ where $\vartheta(X) = \{X_1, \dots, X_n\}$ and $\vartheta(Y) = \{Y_1, \dots, Y_k\}$. That is, φ' equals $\phi(X_1) \wedge \dots \wedge \phi(X_n) \Rightarrow \phi(Y_1) \wedge \dots \wedge \phi(Y_k)$.

We show the *if*-part first. Suppose θ_r makes φ' true. This means, θ_r must make $\phi(X_1) \wedge \dots \wedge \phi(X_n)$ false or $\phi(Y_1) \wedge \dots \wedge \phi(Y_k)$ true. If $\phi(X_1) \wedge \dots \wedge \phi(X_n)$ is false, then for some i between 1 and n we have $\pi_{X_i}^N(t_1) \neq \pi_{X_i}^N(t_2)$, and it follows that r satisfies φ since $\pi_X^N(t_1) \neq \pi_X^N(t_2)$. If $\phi(Y_1) \wedge \dots \wedge \phi(Y_k)$ is true, then $\pi_{Y_j}^N(t_1) = \pi_{Y_j}^N(t_2)$ for all j between 1 and k , and so φ is again satisfied by r since $\pi_Y^N(t_1) = \pi_Y^N(t_2)$ according to Lemma 9 and $Y = Y_1 \sqcup \dots \sqcup Y_k$.

It remains to show the *only if*-part. Suppose $\models_r \varphi$, i.e., $\pi_X^N(t_1) \neq \pi_X^N(t_2)$ or $\pi_Y^N(t_1) = \pi_Y^N(t_2)$. Again, Lemma 9 shows that $\pi_{X_i}^N(t_1) \neq \pi_{X_i}^N(t_2)$ for some i between 1 and n , or $\pi_{Y_j}^N(t_1) = \pi_{Y_j}^N(t_2)$ for all $j = 1, \dots, k$. If $\pi_{X_i}^N(t_1) \neq \pi_{X_i}^N(t_2)$ for some i between 1 and n , then $\theta_r(\phi(X_i)) = \text{false}$ and θ_r satisfies φ' . If $\pi_{Y_j}^N(t_1) = \pi_{Y_j}^N(t_2)$ for all $j = 1, \dots, k$, then $\theta_r(\phi(Y_j)) = \text{true}$ for all $j = 1, \dots, k$. Again, θ_r satisfies φ' .

Let φ be an MVD, say $X \twoheadrightarrow Y$ where $\vartheta(X) = \{X_1, \dots, X_n\}$, $\vartheta(Y) = \{Y_1, \dots, Y_k\}$ and $\vartheta(Y_N^C \dashv X) = \{Z_1, \dots, Z_m\}$. That is, φ' equals $\phi(X_1) \wedge \dots \wedge \phi(X_n) \Rightarrow (\phi(Y_1) \wedge \dots \wedge \phi(Y_k)) \vee (\phi(Z_1) \wedge \dots \wedge \phi(Z_m))$.

We show the *if*-part first. Suppose θ_r makes φ' true. This means, θ_r must make either $\phi(X_1) \wedge \dots \wedge \phi(X_n)$ false, or $\phi(X_1) \wedge \dots \wedge \phi(X_n)$ true and $(\phi(Y_1) \wedge \dots \wedge \phi(Y_k) \text{ true or } \phi(Z_1) \wedge \dots \wedge \phi(Z_m) \text{ true})$. If $\phi(X_1) \wedge \dots \wedge \phi(X_n)$ is false, then for some i between 1 and n we have $\pi_{X_i}^N(t_1) \neq \pi_{X_i}^N(t_2)$, and it follows that $\pi_X^N(t_1) \neq \pi_X^N(t_2)$. Otherwise $\phi(X_1) \wedge \dots \wedge \phi(X_n)$ is true and if $\phi(Y_1) \wedge \dots \wedge \phi(Y_k)$ is true, then $\pi_{Y_j}^N(t_1) = \pi_{Y_j}^N(t_2)$ for all j between 1 and k , and so $\pi_Y^N(t_1) = \pi_Y^N(t_2)$ according to Lemma 9 and $Y = Y_1 \sqcup \dots \sqcup Y_k$. Similarly, if $\phi(Z_1) \wedge \dots \wedge \phi(Z_m)$ is true, then $\pi_{Z_l}^N(t_1) = \pi_{Z_l}^N(t_2)$ since $Y^C \dashv X \leq \bigsqcup_{l=1}^m Z_l$. That is, either $\pi_X^N(t_1) \neq \pi_X^N(t_2)$, or $\pi_X^N(t_1) = \pi_X^N(t_2)$ and $(\pi_Y^N(t_1) = \pi_Y^N(t_2) \text{ or } \pi_{Y^C}^N(t_1) = \pi_{Y^C}^N(t_2))$ holds. The second alternative, according to Lemma 9, is equivalent to $\pi_{X \sqcup Y}^N(t_1) = \pi_{X \sqcup Y}^N(t_2)$ or $\pi_{X \sqcup Y^C}^N(t_1) = \pi_{X \sqcup Y^C}^N(t_2)$. That is, r satisfies $X \twoheadrightarrow Y$.

It remains to show the *only if*-part. Suppose $\models_r \varphi$, i.e., $\pi_X^N(t_1) \neq \pi_X^N(t_2)$, or $\pi_{X \sqcup Y}^N(t_1) = \pi_{X \sqcup Y}^N(t_2)$ or $\pi_{X \sqcup Y^c}^N(t_1) = \pi_{X \sqcup Y^c}^N(t_2)$. If $\pi_X^N(t_1) \neq \pi_X^N(t_2)$, then $\pi_{X_i}^N(t_1) \neq \pi_{X_i}^N(t_2)$ for some i between 1 and n , and $\theta_r(\phi(X_i)) = \text{false}$. That is, θ_r satisfies φ' . If $\pi_{X \sqcup Y}^N(t_1) = \pi_{X \sqcup Y}^N(t_2)$ or $\pi_{X \sqcup Y^c}^N(t_1) = \pi_{X \sqcup Y^c}^N(t_2)$, then $\pi_{Y_j}^N(t_1) = \pi_{Y_j}^N(t_2)$ for all j between 1 and k or $\pi_{Z_l}^N(t_1) = \pi_{Z_l}^N(t_2)$ for all l between 1 and m . Consequently, $\theta_r(\phi(Y_j)) = \text{true}$ for all $j = 1, \dots, k$ or $\theta_r(\phi(Z_l)) = \text{true}$ for all $l = 1, \dots, m$. In any case, θ_r satisfies φ' . \square

6. Reusing relational database design solutions

A direct consequence of the Equivalence theorems from [17,30,31] and the Equivalence theorems 26 and 27 is that relational database design solutions can be applied to problems for databases that are not in first normal form (and vice versa). In this section we will make this correspondence explicit for FDs and MVDs, and use this relationship to provide upper bounds on the time-complexity of the associated implication problems in nested databases.

6.1. Mappings to relation schemata

Let $N \in \mathcal{N}_{\{\text{record}, \text{list}\}}$. We interpret the join-irreducible elements of N as attributes in a relation schema, i.e., the corresponding relation schema of N is $R_N = \mathcal{J}(N)$. Let Σ be a set of FDs and MVDs on N . Each FD $\varphi: X \rightarrow Y$ in Σ is mapped to the relational FD $\varphi': \vartheta(X) \rightarrow \vartheta(Y)$, and each MVD $\varphi: X \twoheadrightarrow Y$ in Σ is mapped to the relational MVD $\varphi': \vartheta(X) \twoheadrightarrow \vartheta(Y)$. Therefore, the corresponding set of relational FDs and MVDs is

$$\Sigma' = \{\sigma' \mid \sigma \in \Sigma\} \cup \{U \rightarrow V \mid U, V \in \mathcal{J}(N), U \text{ covers } V\}.$$

The following result is a direct consequence of Theorem 26 and [30, Theorem 8].

Corollary 35. *Let $N \in \mathcal{N}_{\{\text{record}, \text{list}\}}$, and $\Sigma \cup \{\varphi\}$ be a set of FDs and MVDs on N . Then φ is implied by Σ if and only if φ' is implied by Σ' on R_N . \square*

Example 36. As an example consider again the nested attribute N

HALFTONING(Brightness, INPUT[Level], OUTPUT[Bit])

together with the following set Σ of FDs and MVDs

$$\begin{aligned} \text{HALFTONING}(\text{Brightness}, \text{INPUT}[\lambda], \lambda) &\rightarrow \text{HALFTONING}(\lambda, \text{INPUT}[\text{Level}], \lambda), \\ \text{HALFTONING}(\lambda, \text{INPUT}[\lambda], \lambda) &\rightarrow \text{HALFTONING}(\lambda, \lambda, \text{OUTPUT}[\lambda]), \quad \text{and} \\ \text{HALFTONING}(\lambda, \lambda, \text{OUTPUT}[\lambda]) &\rightarrow \text{HALFTONING}(\lambda, \text{INPUT}[\lambda], \lambda). \end{aligned}$$

The join-irreducibles of N are mapped to flat attribute names as follows:

- HALFTONING(Brightness, λ , λ) is assigned to A ,
- HALFTONING(λ , INPUT[λ], λ) is assigned to B ,
- HALFTONING(λ , INPUT[Level], λ) is assigned to C ,
- HALFTONING(λ , λ , OUTPUT[λ]) is assigned to D , and
- HALFTONING(λ , λ , OUTPUT[Bit]) is assigned to E .

The corresponding set Σ' of FDs and MVDs on the relation schema $R_N = \{A, B, C, D, E\}$ is

$$\Sigma' = \{AB \twoheadrightarrow C, B \rightarrow D, D \rightarrow B, C \rightarrow B, E \rightarrow D\}.$$

In order to determine whether

$$\text{HALFTONING}(\text{Brightness}, \lambda, \text{OUTPUT}[\lambda]) \rightarrow \text{HALFTONING}(\lambda, \lambda, \text{OUTPUT}[\text{Bit}])$$

and

$$\text{HALFTONING}(\text{Brightness}, \lambda, \text{OUTPUT}[\lambda]) \rightarrow \text{HALFTONING}(\lambda, \lambda, \text{OUTPUT}[\text{Bit}])$$

are implied by Σ , respectively, one can check whether $A, D \rightarrow E$ and $A, D \twoheadrightarrow E$ are implied by Σ' , respectively. Well-known techniques from relational databases [6,19] can be applied to compute the dependency basis $DepB(AD) = \{A, B, C, D, E\}$ and closure $(AD)^+ = \{A, D, E\}$ of AD with respect to Σ' . Since $E \notin A^+$ and $E \in DepB(AD)$ it follows that $AD \rightarrow E$ is not implied by Σ' and $AD \twoheadrightarrow E$ is implied by Σ' . Corollary 35 tells us that

$$\text{HALFTONING}(\text{Brightness}, \lambda, \text{OUTPUT}[\lambda]) \rightarrow \text{HALFTONING}(\lambda, \lambda, \text{OUTPUT}[\text{Bit}])$$

is not implied by Σ , but

$$\text{HALFTONING}(\text{Brightness}, \lambda, \text{OUTPUT}[\lambda]) \rightarrow \text{HALFTONING}(\lambda, \lambda, \text{OUTPUT}[\text{Bit}])$$

is indeed implied by Σ . \square

A similar result holds for arbitrary $N \in \mathcal{N}$ and sets Σ of FDs on N . In this case we interpret the extended join-irreducibles of N as attributes in a relation schema, i.e., the corresponding relation schema of N is $R_N = \mathcal{E}(N)$. Each FD $\varphi: \mathcal{X} \rightarrow \mathcal{Y}$ in Σ is mapped to the relational FD $\varphi': \vartheta(\mathcal{X}) \rightarrow \vartheta(\mathcal{Y})$. Therefore, the corresponding set of relational FDs is

$$\Sigma' = \{\sigma' \mid \sigma \in \Sigma\} \cup \{U \rightarrow V \mid U, V \in \mathcal{E}(N), U \text{ covers } V\}.$$

The following result is a direct consequence of Theorem 26 and the Equivalence theorem in [17].

Corollary 37. *Let $N \in \mathcal{N}$, and $\Sigma \cup \{\varphi\}$ be a set of FDs on N . Then φ is implied by Σ if and only if φ' is implied by Σ' on R_N . \square*

6.2. The complexity of related implication problems

We can apply Corollaries 35 and 37 to obtain small upper bounds on the time-complexity of the associated implication problems for the classes of dependencies we have considered. In particular, we obtain smaller bounds than those achieved in previous work [20].

The implication problem for FDs in the presence of records and lists can be solved in time $\mathcal{O}(n)$ where n denotes the total number of join-irreducibles of N that occur in FDs $\vartheta(X) \rightarrow \vartheta(Y)$ in Σ [8]. The implication problem for FDs in the presence of records, lists, sets and multisets can be solved in time $\mathcal{O}(n)$ where n denotes the total number of extended join-irreducibles of N that occur in FDs $\vartheta(\mathcal{X}) \rightarrow \vartheta(\mathcal{Y})$ in Σ [8]. The implication problem for FDs and MVDs in the presence of records and lists can be solved in time $\mathcal{O}(n \log n)$ where n denotes the total number of join-irreducibles of N that occur in FDs $\vartheta(X) \rightarrow \vartheta(Y)$ and MVDs $\vartheta(X) \twoheadrightarrow \vartheta(Y)$ in Σ [19]. Finally, the implication problem for a fragment of Boolean dependencies can be solved in the same time as the corresponding implication problem for the associated class of Boolean propositional formulae, i.e., it is *NP*-complete in the most general case. As our results show, improvements on the time-complexity of deciding implication for these classes of relational dependencies are synonymous with those in nested databases, and vice versa.

Acknowledgement

This research is supported by the Marsden Fund Council from Government funding, administered by the Royal Society of New Zealand.

References

- [1] S. Abiteboul, P. Buneman, D. Suciu, Data on the Web: From Relations to Semistructured Data and XML, Morgan Kaufmann Publishers, 2000.
- [2] S. Abiteboul, R. Hull, V. Vianu, Foundations of Databases, Addison-Wesley, 1995.
- [3] I. Anderson, Combinatorics of Finite Sets, Oxford Science Publications, The Clarendon Press Oxford University Press, New York, 1987.
- [4] T. Asano, N. Katoh, K. Obokata, T. Tokuyama, Matrix rounding under the L_p -discrepancy measure and its application to digital halftoning, SIAM Journal on Computing 32 (6) (2003) 1423–1435.
- [5] M. Atkinson, F. Bancilhon, D. DeWitt, K. Dittrich, D. Maier, S. Zdonik, The object-oriented database system manifesto, in: Proceedings of the International Conference on Deductive and Object-Oriented Databases, 1989, pp. 40–57.
- [6] C. Beeri, On the membership problem for functional and multivalued dependencies in relational databases, Transactions on Database Systems (TODS) 5 (3) (1980) 241–259.

- [7] C. Beeri, A formal approach to object-oriented databases, *Data and Knowledge Engineering* 5 (4) (1990) 353–382.
- [8] C. Beeri, P.A. Bernstein, Computational problems related to the design of normal form relational schemata, *Transactions on Database Systems (TODS)* (1979) 30–59.
- [9] J. Biskup, Achievements of relational database schema design theory revisited, in: *Semantics in Databases*, in: *Lecture Notes in Computer Science*, vol. 1358, Springer, 1998, pp. 29–54.
- [10] T. Bray, J. Paoli, C.M. Sperberg-McQueen, E. Maler, F. Yergeau, Extensible markup language (XML) 1.0 (third edition) W3C recommendation 04 February 2004. <http://www.w3.org/TR/2004/REC-xml-20040204/>, 2004.
- [11] F. Bry, P. Kröger, A computational biology database digest: Data, data analysis, and data management, *Distributed and Parallel Databases* 13 (1) (2003) 7–42.
- [12] P.P. Chen, The entity-relationship model: Towards a unified view of data, *Transactions on Database Systems (TODS)* 1 (1976) 9–36.
- [13] E.F. Codd, A relational model of data for large shared data banks, *Communications of the ACM* 13 (6) (1970) 377–387.
- [14] C. Delobel, Normalisation and hierarchical dependencies in the relational data model, *Transactions on Database Systems (TODS)* 3 (3) (1978) 201–222.
- [15] C. Delobel, M. Adiba, *Relational Database Systems*, North Holland, 1985.
- [16] H. Enderton, *A Mathematical Introduction to Logic*, second edition, Academic Press, 2001.
- [17] R. Fagin, Functional dependencies in a relational data base and propositional logic, *IBM Journal of Research and Development* 21 (6) (1977) 543–544.
- [18] R. Fagin, Multivalued dependencies and a new normal form for relational databases, *Transactions on Database Systems (TODS)* 2 (3) (1977) 262–278.
- [19] Z. Galil, An almost linear-time algorithm for computing a dependency basis in a relational database, *Journal of the ACM* 29 (1) (1982) 96–102.
- [20] S. Hartmann, S. Link, Deciding implication for functional dependencies in complex-value databases, *Theoretical Computer Science* 364 (2) (2006) 212–240.
- [21] S. Hartmann, S. Link, K.-D. Schewe, Axiomatisations of functional dependencies in the presence of records, lists, sets and multisets, *Theoretical Computer Science* 355 (2) (2006) 167–196.
- [22] S. Hartmann, S. Link, K.-D. Schewe, Functional and multivalued dependencies in nested databases generated by record and list constructor, *Annals of Mathematics and Artificial Intelligence* 46 (1–2) (2006) 114–164.
- [23] R. Hull, R. King, *Semantic database modeling: Survey, applications and research issues*, *ACM Computing Surveys* 19 (3) (1987).
- [24] M. Levene, *The Nested Universal Relation Database Model*, Springer, 1992.
- [25] J. Li, S. Ng, L. Wong, Bioinformatics adventures in database research, in: *Proceedings of the International Conference on Database Theory, ICDT*, in: *Lecture Notes in Computer Science*, vol. 2572, Springer, 2002, pp. 31–46.
- [26] J.C.C. McKinsey, A. Tarski, On closed elements in closure algebras, *Annals of Mathematics* 47 (1946) 122–146.
- [27] J. Paredaens, P. De Bra, M. Gyssens, D. Van Gucht, *The Structure of the Relational Database Model*, Springer-Verlag, 1989.
- [28] D.S. Parker, 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.
- [29] Y. Sagiv, An algorithm for inferring multivalued dependencies with an application to propositional logic, *Journal of the ACM* 27 (2) (1980) 250–262.
- [30] Y. Sagiv, C. Delobel, D.S. Parker Jr., R. Fagin, An equivalence between relational database dependencies and a fragment of propositional logic, *Journal of the ACM* 28 (3) (1981) 435–453.
- [31] Y. Sagiv, C. Delobel, D.S. Parker Jr., R. Fagin, Correction to “An equivalence between relational database dependencies and a fragment of propositional logic”, *Journal of the ACM* 34 (4) (1987) 1016–1018.
- [32] K.-D. Schewe, B. Thalheim, Fundamental concepts of object oriented databases, *Acta Cybernetica* 11 (4) (1993) 49–85.
- [33] P. Seshadri, M. Livny, R. Ramakrishnan, The design and implementation of sequence database system, in: *Proceedings of the International Conference on Very Large Data Bases, VLDB*, 1996, pp. 99–110.
- [34] D. Suci, On database theory and XML, *SIGMOD Record* 30 (3) (2001) 39–45.
- [35] B. Thalheim, *Entity-Relationship Modeling: Foundations of Database Technology*, Springer-Verlag, 2000.
- [36] V. Vianu, A web odyssey: From Codd to XML, in: *Principles of Database Systems*, ACM, 2001, pp. 1–15.