Available online at www.sciencedirect.com
SCIENCE DIRECT®
ELSEVIER
Theoretical Computer Science 355 (2006) 167 – 196
Theoretical Computer Science
www.elsevier.com/locate/tcs

Axiomatisations of functional dependencies in the presence of records, lists, sets and multisets☆

Sven Hartmann, Sebastian Link*, Klaus-Dieter Schewe

Information Science Research Centre, Department of Information Systems, Massey University, Private Bag 11222, Palmerston North, New Zealand

Abstract

We investigate functional dependencies in databases that support complex values such as records, lists, sets and multisets. Therefore, an abstract algebraic framework is proposed that classifies data models according to the underlying types they support. This allows to emphasise the impact of the data types rather than the specifics of a particular data model.

The main results are finite, minimal, sound and complete sets of inference rules for the implication of functional dependencies in the presence of records and all combinations of lists, sets and multisets. The inference rules are similar to Armstrong's original axioms for the relational data model, thanks to the algebraic framework. The completeness result, however, requires a deep analysis in the case of sets and, in particular, multisets.

Keywords: Semantics in databases; Data types; Axiomatisation; Functional dependency; Brouwerian algebra

1. Introduction

Functional dependencies (FDs) were introduced in the context of the relational data model (RDM) by Codd in 1972 (see [30]). Such a dependency is defined on some relation schema R and is an expression of the form $X \to Y$ with attribute sets $X, Y \subseteq R$. A relation r over R *satisfies* $X \to Y$ if any two tuples in r that agree on all attributes in X also agree on all attributes in Y. In general, FDs satisfied by some relation over R are not independent from one another. That is, an FD $X \to Y$ is *implied* by a set Σ of FDs, if $X \to Y$ is satisfied by every relation which already satisfies all dependencies in Σ.

If a database designer chooses several FDs to be satisfied by every relation over some relation schema analysed, then *all* implied FDs have to be determined. This allows to gain complete knowledge about all consequences of the semantics defined, and may avoid inconsistencies and undesired behaviour. In practice, however, it is not possible to study all relations and determine whether a dependency is implied by some given set of dependencies. Therefore, one is much more interested in syntactical inference rules which may allow to decide this implication problem. A set $\mathfrak{R}$ of inference rules is called *sound*, if every dependency which can be derived from Σ using only inference rules in $\mathfrak{R}$, is also implied by Σ. In order to capture *all* dependencies derivable from Σ, the set $\mathfrak{R}$ has to be *complete*. That is, every

☆ An extended abstract of this article was presented at the 10th International Workshop on Logic, Language, Information and Computation (WoLLIC), July 2003.

* Corresponding author. Tel.: +64 6 350 5799; fax: +64 6 350 5725.
E-mail addresses: S.Hartmann@massey.ac.nz (S. Hartmann), S.Link@massey.ac.nz (S. Link), K.D.Schewe@massey.ac.nz (K.-D. Schewe).

0304-3975/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2006.01.008

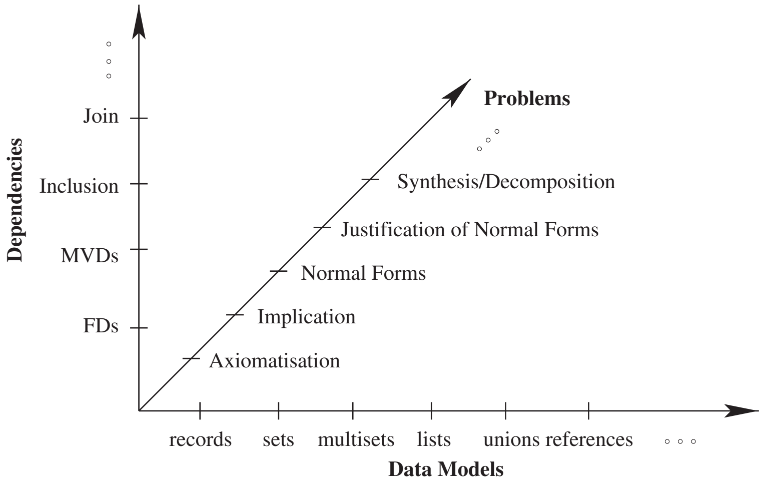

Fig. 1. The three Dimensions of Dependency Theory.

dependency implied by Σ must also be derivable from Σ using only rules in $\mathfrak{R}$. A sound and complete set of inference rules for the implication of FDs in the RDM has been proposed by Armstrong [6,7]. In the context of the RDM such inference rules are easily available, the reason being a well-founded algebraic, yet simple foundation. The set of all attribute sets for some relation schema forms a Boolean algebra with respect to set union, set intersection and set complement. This solid foundation is one of the key reasons for the success of the RDM. On the basis of Armstrong's axiomatisation, polynomial time algorithms for deciding the implication problem [13,17], deciding the equivalence of two given sets of FDs [16] and deriving minimal covers for FDs [59] have been developed. A solution to these problems was a big step towards automated database schema design [16,18] which some researchers see as the ultimate goal in dependency theory [14]. Moreover, normal form proposals such as Boyce–Codd normal form and Third normal form [13,14,18,19,30,31] have been semantically justified a few years later [36,76,82] by formally proving the equivalence to the absence of redundancies and abnormal update behaviour using again Armstrong's axiomatisation.

During the last couple of decades, many new and different data models have been introduced. First, so-called semantic data models have been developed [28,53,72], which were originally just meant to be used as design aids, as application semantics was assumed to be easier captured by these models [10,29,73]. Later on some of these models, especially the nested relational model [64,56], object-oriented and object-relational models [12,40,41,66,67] have become interesting as data models in their own right and some dependency and normalisation theory has been carried over to these advanced data models [23,42–44,49,51,61,63,64,70,83]. Most recently, the major research interest is on the model of semi-structured data and XML [1,24]. Integrity constraints have also been studied in the context of XML [5,26,39,38,79,77]. Almost none of the previous approaches has taken object-equality into consideration when defining constraints, except for a couple of papers that have looked at set equality [42,57]. We believe that object equality is natural and common in real applications and should be included in defining data dependencies.

Several researchers have remarked that classical database design problems need to be revisited in new data formats [4,69,75]. Biskup [21,22] lists in particular two challenges for database design theory: finding a unifying framework and extending achievements to deal with advanced database features such as complex object types. We propose to classify data models according to the type constructors which are supported by the model. This allows to study problems in dependency theory for various classes of dependencies in the presence of various combinations of types, and gives a clear outline of future research, as illustrated by the three dimensions in Fig. 1.

The RDM can be captured by a single application of the record type, arbitrary nesting of record and set type cover aggregation and grouping which are fundamental to many semantic data models as well as the nested RDM [53,56,64]. The Entity-Relationship model and its extensions require record, set and (disjoint) union type [28,72]. A minimal set of types supported by any object-oriented data model includes records, lists, sets and multisets (bags) [8,12,40,41,66,67]. Genomic sequence data models call for support of records, lists and sets [25,58,68]. Finally, XML requires at least record (concatenation), list (Kleene Closure), union (optionality), and reference type [1,24].

In the present paper we consider all combinations of record, set, multiset and list type that include at least the record type, i.e., capture at least the RDM. The need for these various types arises from applications that store ordered relations, time-series data, meteorological and astronomical data streams, runs of experimental data, multidimensional arrays, textual information, voices, sound, images, video, etc. They have been subject to studies in the deductive and temporal database community for some time [62,65], and occur also naturally in object-oriented databases [12,40,41,66] and are in particular important for XML [1,24]. Recently, bioinformatics has become a very important field of research. Of course, lists and sets occur naturally in genomic sequence databases [25,58,68]. Multisets are the fundamental data structure of a number of computational frameworks, such as Gamma coordination language [9], the Chemical Abstract Machine [20], and P systems modelling membrane computing [33]. For a recent survey on the use of multisets in various areas of logic and computer science see [27], in which [54] specifically focuses on database systems. The contributions of this paper are as follows:

- We provide a unifying framework to capture several data models at a time. This allows one to focus on the data types rather than the specifics of a particular data model. Our approach is based on the nesting of flat attributes using record, list, set and multiset constructor. This can be extended to unions, references, etc. It is proven that the set of all subattributes of a fixed nested attribute carries the structure of a Brouwerian algebra (co-Heyting algebra) providing the operations of join $\sqcup$, meet $\sqcap$ and pseudo-difference $\dot{-}$ as generalisations of the standard set operations of the powerset algebra on a relation schema.
- We introduce FDs in the presence of these types, and establish sound inference rules to reason about them. Important differences to the RDM are highlighted.
- The major contributions are finite, sound and complete sets of inference rules for the implication of FDs in the presence of records and all combinations of lists, sets and multisets. The inference rules are very similar to the rules from the RDM, due to the algebraic framework. The presence of the set or multiset type requires two additional axioms which cannot occur in the RDM.
- In fact, the simplicity of the inference rules will allow us to obtain polynomial-time algorithms for deciding the implication of (FDs) in the presence of records, lists, sets, and multisets.
- We study the independence of our inference rules proving that they are indeed minimal in each case. This means that none of the rules can be omitted without losing completeness.
- We compare our approach with previous works, in particular in the context of the nested RDM. It turns out that our class of FDs yields a complementary expressiveness to those classes that have previously been studied.

The paper is organised as follows: Section 2 introduces the abstract data model based on nested attributes which can be obtained from flat attributes by various ways of nesting, i.e., records, sets, multisets and lists. Given a nested attribute N, the set $Sub(N)$ of its subattributes carries the structure of a Brouwerian algebra (co-Heyting Algebra). This is a slightly more general framework than the powerset algebra in the RDM. Section 3 introduces FDs and proposes a generalisation of the well-known Armstrong axioms. In the presence of the set or multiset type, the axiomatisation becomes more sophisticated than in the RDM. This is mainly due to the fact that the values on some subattributes do not, in general, determine the value on the join of those subattributes. Using the algebraic tools it is shown that our generalisation results indeed in a sound and complete set of inference rules for the implication of FDs on nested attributes. In order to show the completeness we construct for each FD σ which is not derivable from the given set Σ of constraints a two element instance that satisfies all the FDs in Σ but which violates σ. This is the standard technique, however, the construction of such a two element instance is non-trivial and involves some combinatorial techniques. The main result of this section provides a finite axiomatisation for FDs in the presence of records, sets, multisets and lists. It is interesting to study whether the inference rules are independent of one another. Section 4 shows that the axiomatisation is indeed minimal, that is, none of the rules can be omitted without losing completeness. Furthermore, we provide minimal axiomatisations for FDs in the context of records and all combinations of lists, sets and multisets. Finally, we compare our approach to work in the literature in Section 6, in particular to works on the nested RDM. We conclude in Section 7 and comment on future work.

2. An abstract data model

The goal of this section is to provide a unifying framework for the study of dependency classes in the context of complex object types. Therefore, we introduce a data model based on the nesting of attributes and subtyping. In this paper, we will deal with records, lists, sets, and multisets. For a survey on complex-valued databases in which the recursive application of record and set constructor are considered see [2].

2.1. Nested attributes

We start with the definition of flat attributes and values for them.

Definition 1. A *universe* is a finite set $\mathcal{U}$ together with domains (i.e. sets of values) $dom(A)$ for all $A \in \mathcal{U}$. The elements of $\mathcal{U}$ are called flat attributes.

For the RDM a universe was sufficient. That is, a relation schema is defined as a finite and non-empty subset $R \subseteq \mathcal{U}$. For data models supporting complex object types, however, nested attributes are needed. In the following definition we use a set $\mathcal{L}$ of labels, and assume that the symbol λ is neither a flat attribute nor a label, i.e., $\lambda \notin \mathcal{U} \cup \mathcal{L}$. Moreover, flat attributes are not labels and vice versa, i.e., $\mathcal{U} \cap \mathcal{L} = \emptyset$.

Definition 2. Let $\mathcal{U}$ be a universe and $\mathcal{L}$ a set of labels. The set $\mathcal{N}A(\mathcal{U}, \mathcal{L})$ of nested attributes over $\mathcal{U}$ and $\mathcal{L}$ is the smallest set satisfying the following conditions:
- $\lambda \in \mathcal{N}A(\mathcal{U}, \mathcal{L})$,
- $\mathcal{U} \subseteq \mathcal{N}A(\mathcal{U}, \mathcal{L})$,
- for $L \in \mathcal{L}$ and $N_1, \ldots, N_k \in \mathcal{N}A(\mathcal{U}, \mathcal{L})$ with $k \geqslant 1$ we have $L(N_1, \ldots, N_k) \in \mathcal{N}A(\mathcal{U}, \mathcal{L})$,
- for $L \in \mathcal{L}$ and $N \in \mathcal{N}A(\mathcal{U}, \mathcal{L})$ we have $L\{N\}, L\langle N\rangle, L[N] \in \mathcal{N}A(\mathcal{U}, \mathcal{L})$.

We call λ null attribute, $L(N_1, \ldots, N_k)$ record-valued attribute, $L\{N\}$ set-valued attribute, $L\langle N\rangle$ multiset-valued attribute, and $L[N]$ list-valued attribute.

From now on we will assume that a universe $\mathcal{U}$ and a set of labels $\mathcal{L}$ are fixed. Instead of writing $\mathcal{N}A(\mathcal{U}, \mathcal{L})$ we simply write $\mathcal{N}A$.

A relation schema $R = \{A_1, \ldots, A_n\}$ can be viewed as the record-valued attribute $R(A_1, \ldots, A_n)$ using the name R as a label. The null attribute λ must not be confused with a null value, which is a distinguished element of a certain domain. The null attribute rather indicates that some information of the underlying nested attribute, i.e., some information on the schema level, has been left out. Further explanations follow.

The mapping dom can be extended from flat to nested attributes, i.e., we define a set $dom(N)$ of values for every nested attribute $N \in \mathcal{N}A$. We denote empty set, empty multiset, and empty list by $\emptyset, \langle\ \rangle, [\ ]$, respectively.

Definition 3. For a nested attribute $N \in \mathcal{N}A$ we define the domain $dom(N)$ as follows:
- $dom(\lambda) = \{ok\}$,
- $dom(A)$ for $A \in \mathcal{U}$ as above,
- $dom(L(N_1, \ldots, N_k)) = \{(v_1, \ldots, v_k) | v_i \in dom(N_i) \text{ for } i = 1, \ldots, k\}$, i.e., the set of all k-tuples $(v_1, \ldots, v_k)$ with $v_i \in dom(N_i)$ for all $i = 1, \ldots, k$,
- $dom(L\{N\}) = \{\{v_1, \ldots, v_n\} | v_i \in dom(N) \text{ for } i = 1, \ldots, n\} \cup \{\emptyset\}$, i.e., $dom(L\{N\})$ is the set of all finite subsets of $dom(N)$,
- $dom(L\langle N\rangle) = \{\langle v_1, \ldots, v_n\rangle | v_i \in dom(N) \text{ for } i = 1, \ldots, n\} \cup \{\langle\ \rangle\}$, i.e., $dom(L\langle N\rangle)$ is the set of all finite multisets with elements in $dom(N)$,
- $dom(L[N]) = \{[v_1, \ldots, v_n] | v_i \in dom(N) \text{ for } i = 1, \ldots, n\} \cup \{[\ ]\}$, i.e., the set of all finite lists with elements in $dom(N)$.

The domain of the record-valued attribute $R(A_1, \ldots, A_n)$ is a set of n-tuples, i.e., an n-ary relation. The value *ok* can be interpreted as the null value "some information exists, but is currently omitted".

2.2. Subattributes

The replacement of flat attribute names by the null attribute λ within a nested attribute decreases the amount of information that is modelled by the corresponding attributes. This fact allows to introduce an order between nested attributes.

Definition 4. The *subattribute relation* $\leqslant$ on the set of nested attributes $\mathcal{N}A$ over $\mathcal{U}$ and $\mathcal{L}$ is defined by the following rules, and the following rules only:

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

For $N, M \in \mathcal{N}A$ we say that M is a *subattribute* of N if and only if $M \leqslant N$ holds. We write $M \not\leqslant N$ if and only if M is not a subattribute of N.

Given the relation schema $R = \{A, B, C\}$, the attribute set $\{A, C\}$ can be viewed as the subattribute $R(A, \lambda, C)$ of the record-valued attribute $R(A, B, C)$. The occurrence of the null attribute λ in $R(A, \lambda, C)$ indicates that the information about the attribute B has been neglected. The inclusion order $\subseteq$ on attribute sets in the RDM is now generalised to the subattribute relation $\leqslant$.

Lemma 5. *The subattribute relation is a partial order on nested attributes.*

Informally, $M \leqslant N$ for $N, M \in \mathcal{N}A$ if and only if M comprises at most as much information as N does. The informal description of the subattribute relation is formally documented by the existence of a projection function $\pi^N_M : dom(N) \rightarrow dom(M)$ in case $M \leqslant N$ holds.

Definition 6. Let $N, M \in \mathcal{N}A$ with $M \leqslant N$. The projection function $\pi^N_M : dom(N) \rightarrow dom(M)$ is defined as follows:

- if $N = M$, then $\pi^N_M = id_{dom(N)}$ is the identity on $dom(N)$,
- if $M = \lambda$, then $\pi^N_\lambda : dom(N) \rightarrow \{ok\}$ is the constant function that maps every $v \in dom(N)$ to ok,
- if $N = L(N_1, \ldots, N_k)$ and $M = L(M_1, \ldots, M_k)$, then $\pi^N_M = \pi^{N_1}_{M_1} \times \cdots \times \pi^{N_k}_{M_k}$ which maps every tuple $(v_1, \ldots, v_k) \in dom(N)$ to $(\pi^{N_1}_{M_1}(v_1), \ldots, \pi^{N_k}_{M_k}(v_k)) \in dom(M)$,
- if $N = L\{N'\}$ and $M = L\{M'\}$, then $\pi^N_M : dom(N) \rightarrow dom(M)$ maps every set $S \in dom(N)$ to the set $\{\pi^{N'}_{M'}(s) : s \in S\} \in dom(M)$,
- if $N = L\langle N' \rangle$ and $M = L\langle M' \rangle$, then $\pi^N_M : dom(N) \rightarrow dom(M)$ maps every multiset $S \in dom(N)$ to the multiset $\langle \pi^{N'}_{M'}(s) : s \in S \rangle \in dom(M)$, and
- if $N = L[N']$ and $M = L[M']$, then $\pi^N_M : dom(N) \rightarrow dom(M)$ maps every list $[v_1, \ldots, v_n] \in dom(N)$ to the list $[\pi^{N'}_{M'}(v_1), \ldots, \pi^{N'}_{M'}(v_n)] \in dom(M)$.

It follows, in particular, that $\emptyset$, $\langle\, \rangle$, $[\,]$ are always mapped to themselves, except when projected on the null attribute λ in which each of them is mapped to ok. Note that for $Y \leqslant X$ we have $\pi^N_Y = \pi^X_Y \circ \pi^N_X$ where $\circ$ denotes the composition of functions.

Example 7. The local dance club keeps record of its classes by storing the date on which the class takes place, the names of its participants, the names of the couples dancing together in that class, and the rating for the class which reflects the average degree of satisfaction of the participants with their dancing partners. In order to capture the semantics we might use the nested attribute

$$N = Dance(Date, Participants\{Name\}, Couple\{Pair(Female, Male)\}, Rating).$$

We will see later on what constraints can be added to improve modelling.

2.3. The Brouwerian algebra of subattributes

Dependency theory in the RDM 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 $-$. Having fixed a nested attribute N one may consider the set $Sub(N)$ of all its subattributes.

Definition 8. Let $N \in \mathcal{N}A$ be a nested attribute. The set $Sub(N)$ of *subattributes* of N is $Sub(N) = \{M | M \leqslant N\}$.

Note that $Sub(N)$ is always finite. Lemma 5 shows that the restriction of $\leqslant$ to $Sub(N)$ is a partial order on $Sub(N)$. We study the algebraic structure of the poset $(Sub(N), \leqslant)$. A *Brouwerian algebra* [60] is a lattice $(L, \sqsubseteq, \sqcup, \sqcap, \dot{-}, 1)$ with top element 1 and a binary operation $\dot{-}$ which satisfies $a \dot{-} b \sqsubseteq c$ iff $a \sqsubseteq b \sqcup c$ for all $c \in L$. In this case, the operation $\dot{-}$ is called the *pseudo-difference*. The *Brouwerian complement* $\neg a$ of $a \in L$ is then defined by $\neg a = 1 \dot{-} a$. A Brouwerian algebra is also called a co-Heyting algebra or a dual Heyting algebra. While in a Heyting algebra the join of an element and its complement is not necessarily the top element, in a Brouwerian algebra the meet of an element and its Brouwerian complement is not necessarily the bottom element. The system of all closed subsets of a topological space is a well-known Brouwerian algebra.

We observe the following: $Sub(\lambda)$ is isomorphic to the Boolean algebra of order 0, $Sub(A)$, A a flat attribute, isomorphic to the Boolean algebra of order 1. $Sub(L(N))$ is isomorphic to $Sub(N)$, $Sub(L(N_1, \ldots, N_n))$ isomorphic to the direct product of $Sub(N_1), \ldots, Sub(N_n)$, and $Sub(L\{N\})$, $Sub(L\langle N\rangle)$, $Sub(L[N])$ are all isomorphic to $Sub(N)$ augmented by a new minimum. It is an easy exercise to show that the set of all (finite) Brouwerian algebras is closed with respect to both operations (add a new minimum, direct product). The following theorem generalises the fact that $(\mathcal{P}(R), \subseteq, \cup, \cap, -, \emptyset, R)$ is a Boolean algebra for a relation schema R in the RDM. Its formal proof consists of verifying the axioms of a Brouwerian algebra.

Theorem 9. $(Sub(N), \leqslant, \sqcup_N, \sqcap_N, \dot{-}_N, N)$ *forms a Brouwerian algebra for every* $N \in \mathcal{N}A$.

Note that $(Sub(N), \leqslant, \sqcup, \sqcap, (\cdot)^{\mathcal{C}}, \lambda, N)$ is in general not Boolean. Take for instance $N = L[A]$ and $Y = L[\lambda]$. Then $Y^{\mathcal{C}} = N$ and $Y \sqcap Y^{\mathcal{C}} = Y \neq \lambda$. Furthermore, $Y^{\mathcal{CC}} = \lambda \neq Y$.

In the following we record some properties for join, meet and pseudo-difference operation on $(Sub(N), \leqslant)$. Obviously, the nested attribute N is the top element of $(Sub(N), \leqslant)$. According to Definition 4 the bottom element λ_N can be described as follows.

Lemma 10. *The bottom element* λ_N *of* $Sub(N)$ *is given by* $\lambda_N = L(\lambda_{N_1}, \ldots, \lambda_{N_k})$ *whenever* $N = L(N_1, \ldots, N_k)$, *and* $\lambda_N = \lambda$ *whenever* N *is not a record-valued attribute*.

Moreover, Definition 4 allows to show the following properties.

Lemma 11. *Let* $N \in \mathcal{N}A$ *and* $X, Y \in Sub(N)$. *The join* $X \sqcup_N Y$, *meet* $X \sqcap_N Y$ *and pseudo-difference* $X \dot{-}_N Y$ *of* X *and* Y *in* $Sub(N)$ *enjoy the following properties*:

- *if* $N = L(N_1, \ldots, N_k)$, $X = L(X_1, \ldots, X_k)$ *and* $Y = L(Y_1, \ldots, Y_k)$, *then* $X \circ_N Y = L(X_1 \circ_{N_1} Y_1, \ldots, X_k \circ_{N_k} Y_k)$ *for* $\circ \in \{\sqcup, \sqcap, \dot{-}\}$
- *if* $N = L\{M\}$, $X = L\{X'\}$, $Y = L\{Y'\}$, *then* $X \circ_N Y = L\{X' \circ_M Y'\}$ for $\circ \in \{\sqcup, \sqcap\}$,
- *if* $X \nleqslant Y$ *and* $N = L\{M\}$, $X = L\{X'\}$, $Y = L\{Y'\}$, *then* $X \dot{-}_N Y = L\{X' \dot{-}_M Y'\}$,
- *if* $N = L\langle M\rangle$, $X = L\langle X'\rangle$, $Y = L\langle Y'\rangle$, *then* $X \circ_N Y = L\langle X' \circ_M Y'\rangle$ for $\circ \in \{\sqcup, \sqcap\}$,
- *if* $X \nleqslant Y$ *and* $N = L\langle M\rangle$, $X = L\langle X'\rangle$, $Y = L\langle Y'\rangle$, *then* $X \dot{-}_N Y = L\langle X' \dot{-}_M Y'\rangle$,
- *if* $N = L[M]$, $X = L[X']$, $Y = L[Y']$, *then* $X \circ_N Y = L[X' \circ_M Y']$ for $\circ \in \{\sqcup, \sqcap\}$, *and*
- *if* $X \nleqslant Y$ *and* $N = L[M]$, $X = L[X']$, $Y = L[Y']$, *then* $X_N Y = L[X'_M Y']$.

In order to simplify notation, occurrences of λ in a record-valued attribute are usually omitted if this does not cause any ambiguities. That is, the subattribute $L(M_1, \ldots, M_k) \leqslant L(N_1, \ldots, N_k)$ is abbreviated by $L(M_{i_1}, \ldots, M_{i_l})$ where

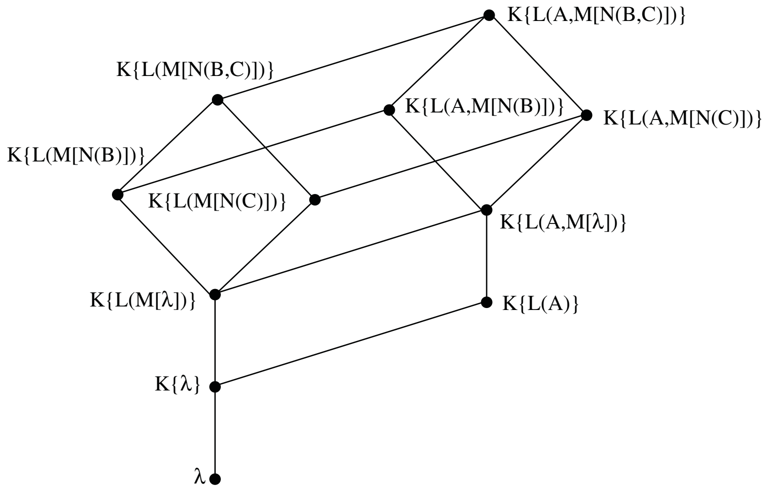

Fig. 2. The Brouwerian algebra of $K\{L(A, M[N(B, C)])\}$.

$\{M_{i_1}, \ldots, M_{i_l}\} = \{M_j : M_j \neq \lambda_{N_j}$ and $1 \leqslant j \leqslant k\}$ and $i_1 < \cdots < i_l$. If $M_j = \lambda_{N_j}$ for all $j = 1, \ldots, k$, then we use λ instead of $L(M_1, \ldots, M_k)$. The subattribute $L_1(A, \lambda, L_2[L_3(\lambda, \lambda)])$ of $L_1(A, B, L_2[L_3(C, D)])$ is abbreviated by $L_1(A, L_2[\lambda])$. However, the subattribute $L(A, \lambda)$ of $L(A, A)$ cannot be abbreviated by $L(A)$ since this may also refer to $L(\lambda, A)$.

If the context allows, we omit the index N from the operations $\sqcup_N, \sqcap_N, \dot{-}_N$ and from λ_N. The Brouwerian algebra for $K\{L(A, M[N(B, C)])\}$ is illustrated in Fig. 2.

Given some nested attribute $N \in \mathcal{N}A$ and $Y, Z \in Sub(N)$, we use $Y_N^{\mathcal{C}} = N \dot{-} Y$ to denote the *Brouwerian complement* of Y in $Sub(N)$. Again, we omit the subscript N if the context allows. The pseudo-difference $Z \dot{-} Y$ of Z and Y in $Sub(N)$ satisfies $Z \dot{-} Y \leqslant X$ if and only if $Z \leqslant Y \sqcup X$ for all $X \in Sub(N)$. Consequently, for all $X \in Sub(N)$ holds $Y^{\mathcal{C}} \leqslant X$ if and only if $X \sqcup Y = N$ holds.

2.4. Order, multiplicity and the null attribute

Elements of a list are totally ordered and the same element may occur several times. Elements of a multiset are not ordered, but the same element may still occur several times. The elements of a set are not ordered and distinct, i.e., an element of a set occurs precisely once.

We give some more explanations on the null attribute λ. From an algebraic point of view it is simply the bottom element $N \dot{-} N$ of the Brouwerian algebra carried by N. As already seen, replacing occurrences of nested attributes by the null attribute according to the rules of the subattribute relationship results in a subattribute and therefore in a decrease of the amount of information that can be modelled. The null attribute therefore allows to obtain different layers of information generating ultimately the structure of a Brouwerian algebra for a fixed database schema.

However, the null attribute also offers some interesting features for database modelling, depending on the presence of certain complex objects. Consider for instance the nested attribute *Shopping*(*Person*, *Purchase*[*Article*]) which is used to store the list of articles purchased by a person. Two elements from the corresponding domain could be (*Toni*, [*Shoes*, *Top*, *Shoes*, *Jacket*]) and (*Sebastian*, [ ]). The projections of these elements on the subattribute *Shopping*(*Person*, *Purchase*[λ]) are (*Toni*, [*ok*, *ok*, *ok*, *ok*]) and (*Sebastian*, [ ]) still revealing that Toni bought 4 articles and Sebastian none. Suppose that instead of using the list-valued attribute *Purchase*[*Article*] we used a set-valued attribute *Purchase*{*Article*}, i.e., we are only interested in the different articles a person buys, and not in the order nor the number of the same articles. The element (*Toni*, {*Shoes*, *Top*, *Jacket*}) is mapped to (*Toni*, {*ok*}), and the element (*Sebastian*, ∅) is mapped to itself. The subattribute *Shopping*(*Person*, *Purchase*{λ}) therefore reveals whether a person bought anything at all. The feature to store the same data repeatedly therefore enables *counting*.

The second feature is the ability to model order. This property implies that the projections of any tuple on two subattributes X and Y of N always determine the projection of that tuple on the join $X \sqcup Y$. In case of the set or multiset constructor, this property is not valid anymore. This will be demonstrated in Example 14.

3. Axiomatising functional dependencies

We define FDs on a nested attribute and introduce some sound inference rules for the implication of FDs.

Definition 12. Let $N \in \mathcal{N}A$ be a nested attribute. A functional dependency (FD) on N is an expression of the form $\mathcal{X} \to \mathcal{Y}$ where $\mathcal{X}, \mathcal{Y} \subseteq Sub(N)$ are non-empty. A set $r \subseteq dom(N)$ satisfies the FD $\mathcal{X} \to \mathcal{Y}$ on N, denoted by $\models_r \mathcal{X} \to \mathcal{Y}$, if and only if $\pi_Y^N(t_1) = \pi_Y^N(t_2)$ holds for all $Y \in \mathcal{Y}$ whenever $\pi_X^N(t_1) = \pi_X^N(t_2)$ holds for all $X \in \mathcal{X}$ and any $t_1, t_2 \in r$.

In case a set of subattributes is the singleton $\{X\}$ we also write X instead.

Example 13. Consider Example 7 again. We first list FDs that should be specified for this application. The FD

$$Dance(Date) \to Dance(Participants\{Name\}, Couples\{Pair(Female, Male)\}, Rating)$$

says informally that the date on which the class takes place determines the names of its participants, the couples which dance together and the rating of this class. The FD

$$Dance(Participants\{Name\}) \to \{Dance(Couples\{Pair(Female)\}), Dance(Couples\{Pair(Male)\})\}$$

tells us that the set of participants determines the set of participating females, and the set of participating males. The FD

$$\{Dance(Couples\{Pair(Female)\}), Dance(Couples\{Pair(Male)\})\} \to Dance(Participants\{Name\})$$

says that the sets of participating females and participating males determines the set of participants. Finally, the rating of each class is determined by the couples that dance together, i.e.,

$$Dance(Couples\{Pair(Female, Male)\}) \to Dance(Rating).$$

Examples of FDs which should not be specified for this application are the following.

$$Dance(Participants\{Name\}) \to Dance(Couples\{Pair(Female, Male)\})$$

is not a reasonable constraint for this application since dance partners may switch from class to class. Neither are the FDs

$$Dance(Participants\{Name\}) \to Dance(Rating) \text{ and}$$
$$\{Dance(Couples\{Pair(Female)\}), Dance(Couples\{Pair(Male)\})\} \to Dance(Rating)$$

meaningful since the rating of the class is not determined by the participants themselves, but by the combination of dance partners.

The notions of implication ($\models$) and derivability ($\vdash_{\mathfrak{R}}$) with respect to a rule system $\mathfrak{R}$ for FDs on a nested attribute can be defined analogously to the notions in the RDM (see for instance [2, pp. 163–168]). Let Σ be a set of FDs, and $\mathcal{X} \to \mathcal{Y}$ an FD on some nested attribute N. Real-life databases are inherently finite. Therefore, our attention should be firstly directed towards the finite implication problem where $\Sigma \models_f \mathcal{X} \to \mathcal{Y}$ holds whenever any *finite* instance $r \subseteq dom(N)$ that satisfies all FDs in Σ also satisfies $\mathcal{X} \to \mathcal{Y}$. However, in the case of FDs the finite implication problem coincides with the unrestricted implication problem $\Sigma \models \mathcal{X} \to \mathcal{Y}$. It is obvious that $\models \subseteq \models_f$ holds. If there is an infinite $r \subseteq dom(N)$ with $\models_r \Sigma$ and $\not\models_r \mathcal{X} \to \mathcal{Y}$, then there are $t_1, t_2 \in r$ with $\not\models_{\{t_1,t_2\}} \mathcal{X} \to \mathcal{Y}$. However, $\models_{\{t_1,t_2\}} \Sigma$ follows directly from $\models_r \Sigma$. It follows that also $\models_f \subseteq \models$ holds, i.e., unrestricted and finite implication coincide. We are interested in the set of all FDs implied by Σ, i.e., $\Sigma^* = \{\varphi \mid \Sigma \models \varphi\}$. Our aim is finding a set $\mathfrak{R}$ of inference rules which is *sound* ($\Sigma^+_{\mathfrak{R}} \subseteq \Sigma^*$) and *complete* ($\Sigma^* \subseteq \Sigma^+_{\mathfrak{R}}$), where $\Sigma^+_{\mathfrak{R}} = \{\varphi \mid \Sigma \vdash_{\mathfrak{R}} \varphi\}$ is the set of FDs derivable from Σ

using only inference rules from $\mathfrak{R}$. The following example reveals a fundamental difference between sound inference rules in the RDM and our abstract data model. That is, the FD $X \to X \sqcup Y$ is no longer implied by the FD $X \to Y$ in the presence of sets or multisets. That means, the projections $\pi_X^N(t)$ and $\pi_Y^N(t)$ of a tuple $t \in dom(N)$ on subattributes $X, Y \in Sub(N)$ do not determine the projection $\pi_{X \sqcup Y}^N(t)$ on the join $X \sqcup Y$.

Example 14. Consider Example 7. We choose $r = \{t_1, t_2\} \subseteq dom(N)$ with

t_1=(29.2.1600, {*Dulcinea*, *Don Quixote*, *Theresa*, *Sancho*}, {(*Dulcinea*, *Don Quixote*), (*Theresa*, *Sancho*)}, 10) and

t_2=(1.3.1600, {*Dulcinea*, *Don Quixote*, *Theresa*, *Sancho*}, {(*Dulcinea*, *Sancho*), (*Theresa*, *Don Quixote*)}, 3).

The projections of t_1 and t_2 on $X = Dance(Couples\{Pair(Female)\})$ are both

$$(ok, ok, \{(Dulcinea, ok), (Theresa, ok)\}, ok)$$

and the projections of t_1 and t_2 on $Y = Dance(Couples\{Pair(Male)\})$ are both

$$(ok, ok, \{(ok, Sancho), (ok, Don\ Quixote)\}, ok).$$

However, the projection of t_1 on the join $Dance(Couples\{Pair(Female, Male)\})$ of X and Y is

$$(ok, ok, \{(Dulcinea, Don\ Quixote), (Theresa, Sancho)\}, ok).$$

This is different from the projection of t_2 on $Dance(Couples\{Pair(Female, Male)\})$ which is

$$(ok, ok, \{(Dulcinea, Sancho), (Theresa, Don\ Quixote)\}, ok).$$

Therefore, the projections $\pi_X^N(t)$ and $\pi_Y^N(t)$ of a tuple $t \in dom(N)$ on subattributes $X, Y \in Sub(N)$ do not determine the projection $\pi_{X \sqcup Y}^N(t)$ on the join $X \sqcup Y$.

Before we introduce some inference rules for FDs, we will give a sufficient condition when values on subattributes X and Y do determine the values on $X \sqcup Y$.

Definition 15. Let $N \in \mathcal{NA}$. The subattributes $X, Y \in Sub(N)$ are reconcilable if and only if one of the following conditions is satisfied:

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

Given $X, Y \in Sub(N)$ that are reconcilable and some $t \in dom(N)$ the projections $\pi_X^N(t)$ and $\pi_Y^N(t)$ determine $\pi_{X \sqcup Y}^N(t)$.

Lemma 16. *Let $N \in \mathcal{NA}$, $X, Y \in Sub(N)$ reconcilable and $t_1, t_2 \in dom(N)$. If $\pi_X^N(t_1) = \pi_X^N(t_2)$ and $\pi_Y^N(t_1) = \pi_Y^N(t_2)$, then $\pi_{X \sqcup Y}^N(t_1) = \pi_{X \sqcup Y}^N(t_2)$.*

Proof. We proceed by induction on the structure of N. If $Y \leqslant X$, then $X \sqcup Y = X$ and the statement follows from the assumption that $\pi_X^N(t_1) = \pi_X^N(t_2)$. If $X \leqslant Y$, then $X \sqcup Y = Y$ and the statement follows from the assumption that $\pi_Y^N(t_1) = \pi_Y^N(t_2)$. Let $N = L(N_1, \ldots, N_k)$, $X = L(X_1, \ldots, X_k)$ and $Y = L(Y_1, \ldots, Y_k)$. Consequently, $t_1, t_2 \in dom(N)$ have the form $t_1 = (t_1^1, \ldots, t_k^1)$ and $t_2 = (t_1^2, \ldots, t_k^2)$ with $t_j^i \in dom(N_j)$ for $j = 1, \ldots, k$ and $i = 1, 2$. From $\pi_X^N(t_1) = \pi_X^N(t_2)$ follows $\pi_{X_i}^{N_i}(t_i^1) = \pi_{X_i}^{N_i}(t_i^2)$ for $i = 1, \ldots, k$ by definition of the projection function. Similarly follows $\pi_{Y_i}^{N_i}(t_i^1) = \pi_{Y_i}^{N_i}(t_i^2)$ for $i = 1, \ldots, k$ from $\pi_Y^N(t_1) = \pi_Y^N(t_2)$. The assumption that X and Y are reconcilable implies that X_i and Y_i are reconcilable for all $i = 1, \ldots, k$. Consequently, we conclude $\pi_{X_i \sqcup Y_i}^{N_i}(t_i^1) = \pi_{X_i \sqcup Y_i}^{N_i}(t_i^2)$ for $i = 1, \ldots, k$.

This shows that

$$\begin{aligned}\pi^N_{X\sqcup Y}(t_1) &= (\pi^{N_1}_{X_1\sqcup Y_1}(t^1_1),\ldots,\pi^{N_k}_{X_k\sqcup Y_k}(t^1_k))\\ &= (\pi^{N_1}_{X_1\sqcup Y_1}(t^2_1),\ldots,\pi^{N_k}_{X_k\sqcup Y_k}(t^2_k))\\ &= \pi^N_{X\sqcup Y}(t_2)\end{aligned}$$

which we had to prove. It remains to consider the case where $N = L[N']$, $X = L[X']$, $Y = L[Y']$. Consequently, $t_1, t_2 \in dom(N)$ have the form $t_1 = [t^1_1,\ldots,t^1_k]$ and $t_2 = [t^2_1,\ldots,t^2_l]$ with $t^1_i, t^2_j \in dom(N')$ for $i = 1,\ldots,k$ and $j = 1,\ldots,l$. From $\pi^N_X(t_1) = \pi^N_X(t_2)$ follows $k = l$ and $\pi^{N'}_{X'}(t^1_i) = \pi^{N'}_{X'}(t^2_i)$ for $i = 1,\ldots,k$ by definition of the projection function. Similarly follows $\pi^{N'}_{Y'}(t^1_i) = \pi^{N'}_{Y'}(t^2_i)$ for $i = 1,\ldots,k$ from $\pi^N_Y(t_1) = \pi^N_Y(t_2)$. The assumption that X and Y are reconcilable implies that X' and Y' are reconcilable. Consequently, we conclude $\pi^{N'}_{X'\sqcup Y'}(t^1_i) = \pi^{N'}_{X'\sqcup Y'}(t^2_i)$ for $i = 1,\ldots,k$. This shows that

$$\begin{aligned}\pi^N_{X\sqcup Y}(t_1) &= [\pi^{N'}_{X'\sqcup Y'}(t^1_1),\ldots,\pi^{N'}_{X'\sqcup Y'}(t^1_k)]\\ &= [\pi^{N'}_{X'\sqcup Y'}(t^2_1),\ldots,\pi^{N'}_{X'\sqcup Y'}(t^2_k)]\\ &= \pi^N_{X\sqcup Y}(t_2)\end{aligned}$$

which we had to prove. If N is a set-valued or multiset-valued attribute, then $X \leqslant Y$ or $Y \leqslant X$ according to Definition 15 of reconcilable subattributes. □

We will see later on that this condition is exact, i.e. if the values on X and Y do determine the value on $X \sqcup Y$, then X and Y are necessarily reconcilable.

Definition 17. The following inference rules

$$\frac{}{\mathcal{X} \to \mathcal{Y}}\ \mathcal{Y} \subseteq \qquad \frac{\mathcal{X} \to \mathcal{Y}}{\mathcal{X} \to \mathcal{X} \cup \mathcal{Y}},$$

(reflexivity axiom) (subattribute axiom) (extension rule)

$$\frac{}{\{X, Y\} \to \{X \sqcup_N Y\}}\ X, Y \text{ reconcilable}, \qquad \frac{\mathcal{X} \to \mathcal{Y},\ \mathcal{Y} \to \mathcal{Z}}{\mathcal{X} \to \mathcal{Z}}$$

(restricted join axiom) (transitivity rule)

are called the generalised Armstrong axioms for FDs.

3.1. Soundness and some useful inference rules

We show that all FDs that can be derived from a given set Σ of FDs using any of the rules from Definition 17 are also implied by Σ.

Proposition 18. *The generalised Armstrong axioms for FDs are sound.*

Proof. Let $N \in \mathcal{N}A$ and $r \subseteq dom(N)$. First consider the reflexivity axiom, and let $t_1, t_2 \in r$ with $\pi^N_X(t_1) = \pi^N_X(t_2)$ for all $X \in \mathcal{X}$. Since $\mathcal{Y} \subseteq \mathcal{X}$ this implies that $\pi^N_Y(t_1) = \pi^N_Y(t_2)$ holds also for all $Y \in \mathcal{Y}$.

For the subattribute axiom let again $t_1, t_2 \in r$ with $\pi^N_X(t_1) = \pi^N_X(t_2)$. For $Y \leqslant X$ follows $\pi^N_Y = \pi^X_Y \circ \pi^N_X$ where $\circ$ denotes the composition of functions. Consequently, $\pi^N_Y(t_1) = \pi^X_Y(\pi^N_X(t_1)) = \pi^X_Y(\pi^N_X(t_2)) = \pi^N_Y(t_2)$.

In order to prove the extension rule let $t_1, t_2 \in r$ with $\pi^N_X(t_1) = \pi^N_X(t_2)$ for all $X \in \mathcal{X}$. Since $\models_r \mathcal{X} \to \mathcal{Y}$ holds, it follows that $\pi^N_Y(t_1) = \pi^N_Y(t_2)$ holds for all $Y \in \mathcal{Y}$. Consequently, $\pi^N_Z(t_1) = \pi^N_Z(t_2)$ is true for all $Z \in \mathcal{X} \cup \mathcal{Y}$.

For the restricted join axiom let X and Y be reconcilable, and $r \subseteq dom(N)$. Let $t_1, t_2 \in r$ with $\pi^N_X(t_1) = \pi^N_X(t_2)$ and $\pi^N_Y(t_1) = \pi^N_Y(t_2)$. Lemma 16 shows that $\pi^N_{X\sqcup Y}(t_1) = \pi^N_{X\sqcup Y}(t_2)$ holds as well. The correctness of the restricted join axiom follows.

For the proof of the transitivity rule let $t_1, t_2 \in r$ with $\pi_X^N(t_1) = \pi_X^N(t_2)$ for all $X \in \mathcal{Y}$. Since $\models_r \mathcal{X} \to \mathcal{Y}$ holds, we infer $\pi_Y^N(t_1) = \pi_Y^N(t_2)$ for all $Y \in \mathcal{Y}$. Moreover, $\models_r \mathcal{Y} \to \mathcal{Z}$ which implies $\pi_Z^N(t_1) = \pi_Z^N(t_2)$ for all $Z \in \mathcal{Z}$. This proves that $\models_r \mathcal{X} \to \mathcal{Z}$ holds as well. □

Recall that the famous Armstrong axioms for the implication of FDs in the RDM consist of the reflexivity axiom, the extension rule and the transitivity rule with $\mathcal{X}, \mathcal{Y}$ and $\mathcal{Z}$ being sets of flat attribute names. The subattribute and restricted join axioms, however, are not needed in the RDM since flat attribute names are not comparable anyway, i.e., form an anti-chain. We derive a couple of sound inference rules from the generalised Armstrong axioms which will be needed in the completeness proof.

Proposition 19. *The following rules*

$$\underset{(\lambda\text{-axiom})}{\frac{}{\mathcal{X} \to \{\lambda\}}} \qquad \underset{(\textit{union rule})}{\frac{\mathcal{X} \to \mathcal{Y}, \mathcal{X} \to \mathcal{Z}}{\mathcal{X} \to \mathcal{Y} \cup \mathcal{Z}}} \qquad \underset{(\textit{subattribute rule})}{\frac{\mathcal{X} \to \{Z\}}{\mathcal{X} \to \{Y\}}\; Y \leqslant Z} \qquad \underset{(\textit{subset rule})}{\frac{\mathcal{X} \to \mathcal{Z}}{\mathcal{X} \to \mathcal{Y}}\; \mathcal{Y} \subseteq \mathcal{Z}}$$

can be derived from the generalised Armstrong axioms, and are thus sound.

Proof. The following derivation trees show that each inference rule is derivable from the generalised Armstrong axioms.

λ-axiom: The set $\mathcal{X}$ is non-empty, say $X \in \mathcal{X}$.

$$\frac{\frac{}{\mathcal{X} \to \{X\}}\{X\} \subseteq \mathcal{X} \qquad \frac{}{\{X\} \to \{\lambda\}}\lambda \leqslant X}{\mathcal{X} \to \{\lambda\}}$$

union rule:

$$\frac{\dfrac{\mathcal{X} \to \mathcal{Y}}{\mathcal{X} \to \mathcal{X} \cup \mathcal{Y}} \qquad \dfrac{\dfrac{\dfrac{\frac{}{\mathcal{X} \cup \mathcal{Y} \to \mathcal{X}}\mathcal{X} \subseteq \mathcal{X} \cup \mathcal{Y} \quad \mathcal{X} \to \mathcal{Z}}{\mathcal{X} \cup \mathcal{Y} \to \mathcal{Z}}}{\mathcal{X} \cup \mathcal{Y} \to \mathcal{X} \cup \mathcal{Y} \cup \mathcal{Z}} \qquad \dfrac{}{\mathcal{X} \cup \mathcal{Y} \cup \mathcal{Z} \to \mathcal{Y} \cup \mathcal{Z}}}{\mathcal{X} \cup \mathcal{Y} \to \mathcal{Y} \cup \mathcal{Z}}}{\mathcal{X} \to \mathcal{Y} \cup \mathcal{Z}}$$

subattribute rule:

$$\frac{\mathcal{X} \to \{Z\} \qquad \frac{}{\{Z\} \to \{Y\}}Y \leqslant Z}{\mathcal{X} \to \{Y\}}$$

subset rule:

$$\frac{\mathcal{X} \to \mathcal{Z} \qquad \frac{}{\mathcal{Z} \to \mathcal{Y}}\mathcal{Y} \subseteq \mathcal{Z}}{\mathcal{X} \to \mathcal{Y}}$$

The soundness of each inference rule follows therefore from the derivability from the generalised Armstrong axioms. □

3.2. Completeness

We will use this section to prove the completeness of the generalised Armstrong axioms for the implication of FDs in the presence of records, lists, sets, and multisets. The key idea for the completeness proof follows the original lines of reasoning: for every $\mathcal{X} \to \mathcal{Y} \notin \Sigma^+$ a two element instance $\{t_1, t_2\}$ is constructed which satisfies all FDs in Σ, but does not satisfy $\mathcal{X} \to \mathcal{Y}$. In fact, the projections of t_1 and t_2 will coincide on exactly those subattributes which are in the closure $\mathcal{X}^+ = \{Z : \mathcal{X} \to \{Z\} \in \Sigma^+\}$ of $\mathcal{X}$ with respect to Σ. The main difficulty of the proof is the construction of such a two element instance which is particularly difficult for sets and multisets.

The proof is divided into five parts. First, we show the completeness in Theorem 20 utilising the fact that the closure $\mathcal{X}^+$ is a non-empty ideal that is closed under the join of reconcilable attributes. Recall that an ideal [3,34] of some poset $(S, \leqslant)$ is a subset $\mathcal{I} \subseteq S$ which is closed downwards with respect to $\leqslant$, i.e., if $X \in \mathcal{I}$ and $Y \leqslant X$, then $Y \in \mathcal{I}$ as well.

In order to complete the proof of Theorem 20 it remains to construct a two element instance $\{t_1, t_2\}$ such that the projections of t_1 and t_2 coincide on exactly those subattributes which belong to a non-empty ideal that is closed under the join of reconcilable elements. The second part of the proof is Lemma 21 where the two elements t_1 and t_2 are inductively constructed for null, flat, record- and list-valued attributes. The third part of the proof consists of technical definitions and lemmata in order to deal with the remaining cases. Part four shows the construction for set-valued attributes in Lemma 25, and the final part considers multiset-valued attributes in Lemma 29. In each part, the construction is illustrated by examples.

3.2.1. The main theorem

Theorem 20. *The generalised Armstrong axioms are sound and complete for the implication of FDs in the presence of records, lists, sets and multisets.*

Proof. Soundness has been established in Proposition 18. We show the completeness. Let $N \in \mathcal{N}A$ and Σ be a set of FDs on N. Let $\mathcal{X} \to \mathcal{Y}$ be an FD on N with $\mathcal{X} \to \mathcal{Y} \notin \Sigma^+$. Let $\mathcal{X}^+ = \{Z : \mathcal{X} \to \{Z\} \in \Sigma^+\}$ be the closure of $\mathcal{X}$ with respect to Σ. Then $\lambda \in \mathcal{X}^+$ according to the λ-axiom. The derivability of the union rule implies that $\mathcal{X} \to \mathcal{X}^+ \in \Sigma^+$ holds. If $\mathcal{Y}$ was a subset of $\mathcal{X}^+$, the subset rule would imply that $\mathcal{X} \to \mathcal{Y} \in \Sigma^+$, a contradiction to our assumption. Hence, $\mathcal{Y} \nsubseteq \mathcal{X}^+$, i.e., there is some $Z \in \mathcal{Y}$ with $Z \notin \mathcal{X}^+$. According to the subattribute rule $\mathcal{X}^+$ is an ideal with respect to $\leqslant$. Moreover, if $U, V \in \mathcal{X}^+$ are reconcilable, then the restricted join axiom implies that $U \sqcup V \in \mathcal{X}^+$, too. Therefore, using Lemma 21 we define $r = \{t_1, t_2\} \subseteq dom(N)$ by

$$\pi^N_W(t_1) = \pi^N_W(t_2) \quad \text{if and only if} \quad W \in \mathcal{X}^+ \tag{1}$$

holds. It is immediate that $\not\models_r \mathcal{X} \to \{Z\}$, and this implies $\not\models_r \mathcal{X} \to \mathcal{Y}$ by definition. It remains to show that $\models_r \Sigma$. Therefore, take any $\mathcal{U} \to \mathcal{V} \in \Sigma$.

- If $\mathcal{U} \nsubseteq \mathcal{X}^+$, then $\pi^N_U(t_1) \neq \pi^N_U(t_2)$ for some $U \in \mathcal{U}$ by (1). Obviously, $\models_r \mathcal{U} \to \mathcal{V}$.
- If $\mathcal{U} \subseteq \mathcal{X}^+$, then $\pi^N_U(t_1) = \pi^N_U(t_2)$ for all $U \in \mathcal{U}$ by (1). Since $\mathcal{X} \to \mathcal{X}^+ \in \Sigma^+$ it follows from the subset rule that also $\mathcal{X} \to \mathcal{U} \in \Sigma^+$ holds. Applying the transitivity rule again results in $\mathcal{X} \to \mathcal{V} \in \Sigma^+$. Hence $\mathcal{V} \subseteq \mathcal{X}^+$ by definition of the closure $\mathcal{X}^+$. We conclude by (1) that $\pi^N_V(t_1) = \pi^N_V(t_2)$ holds for all $V \in \mathcal{V}$. This shows $\models_r \mathcal{U} \to \mathcal{V}$.

As $\Sigma^* = \{\mathcal{X} \to \mathcal{Y} | \Sigma \models \mathcal{X} \to \mathcal{Y}\}$, it follows that $\models_r \Sigma^*$. Therefore, $\mathcal{X} \to \mathcal{Y} \notin \Sigma^*$. This proves the completeness. □

3.2.2. The main lemma

The main lemma uses induction arguments for record- and list-valued attributes leaving the cases of set- and multiset-valued attributes for later.

Lemma 21. *Let $N \in \mathcal{N}A$, and $\emptyset \neq \mathcal{X} \subseteq Sub(N)$ an ideal with respect to $\leqslant$ with the property that for reconcilable $X, Y \in \mathcal{X}$ also $X \sqcup Y \in \mathcal{X}$ holds. Then there are $t_N, t'_N \in dom(N)$ with $\pi^N_W(t_N) = \pi^N_W(t'_N)$ if and only if $W \in \mathcal{X}$.*

Proof. The proof is done by induction on N. The case $N = \lambda$ is trivial. If $N = A$ is a flat attribute, then there are two cases $\mathcal{X} = \{\lambda\}$ and $\mathcal{X} = \{\lambda, A\}$ to consider. In the first case we choose $t_A = a$, $t'_A = a'$ with $a, a' \in dom(A)$ and $a \neq a'$, in the second case $t_A = a = t'_A$.

Consider now the case where $N = L(N_1, \ldots, N_k)$. For every $X \in \mathcal{X}$ we have $X = (X \sqcap L(N_1)) \sqcup \cdots \sqcup (X \sqcap L(N_k))$. Consequently, $\mathcal{X}_i = \{X \sqcap L(N_i) : X \in \mathcal{X}\}$ is a non-empty ideal in $Sub(L(N_i))$ for every $i = 1, \ldots, k$. Let $X_i, Y_i \in \mathcal{X}_i$ be reconcilable. Then $X_i = X \sqcap L(N_i)$ and $Y_i = Y \sqcap L(N_i)$ for some $X, Y \in \mathcal{X}$. Since $\mathcal{X}$ is an ideal it follows from $X_i \leqslant X$ and $Y_i \leqslant Y$ that $X_i, Y_i \in \mathcal{X}$, too. We conclude that $X_i \sqcup Y_i \in \mathcal{X}$ since $\mathcal{X}$ is closed under the join of reconcilable elements. Since $X_i \sqcup Y_i = (X \sqcup Y) \sqcap L(N_i) \in \mathcal{X}$ it follows that $(X_i \sqcup Y_i) \sqcap L(N_i) = X_i \sqcup Y_i \in \mathcal{X}_i$ by definition of $\mathcal{X}_i$. That is, $\mathcal{X}_i$ is also closed under the join of reconcilable elements. We know by hypothesis that for all $i = 1, \ldots, k$ there are $t_{L(N_i)}, t'_{L(N_i)} \in dom(L(N_i))$ with $\pi^{L(N_i)}_{L(W_i)}(t_{L(N_i)}) = \pi^{L(N_i)}_{L(W_i)}(t'_{L(N_i)})$ if and only if $L(W_i) \in \mathcal{X}_i$ holds. Now we choose $t_N = (t_{L(N_1)}, \ldots, t_{L(N_k)})$ and $t'_N = (t'_{L(N_1)}, \ldots, t'_{L(N_k)})$ and have the equivalence of $\pi^N_W(t_N) = \pi^N_W(t'_N)$ if and only if $W \in \mathcal{X}$ with $\pi^{L(N_i)}_{L(W_i)}(t_{L(N_i)}) = \pi^{L(N_i)}_{L(W_i)}(t'_{L(N_i)})$ if and only if $L(W_i) \in \mathcal{X}_i$ holds for $i = 1, \ldots, k$.

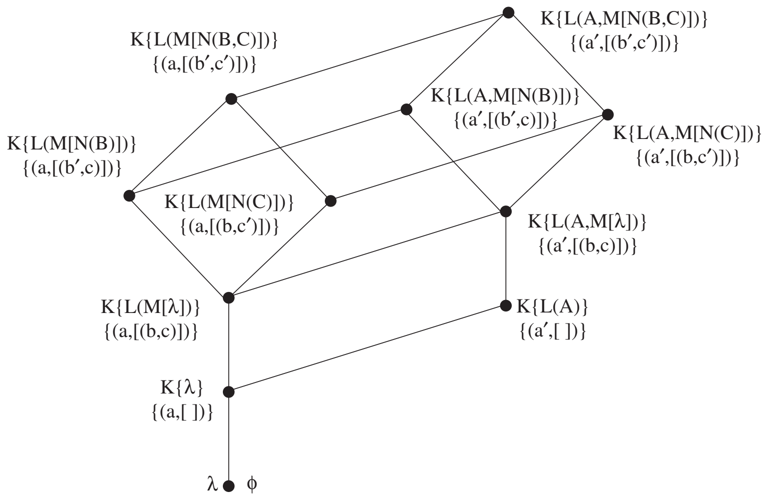

Fig. 3. Identifying terms of the Algebra $K\{L(A, M[N(B, C)])\}$.

Suppose $N = L[N']$. Then $\mathcal{X} = \{L[M] : M \in \mathcal{Y}\} \cup \{\lambda\}$ for an ideal $\mathcal{Y} \subseteq Sub(N')$. If $\mathcal{Y} = \emptyset$, then $\mathcal{X} = \{\lambda\}$. Define $t_N = [\,]$, $t'_N = [n'] \in dom(N)$ for some $n' \in dom(N')$. For $\lambda \neq W \in Sub(N)$, say $W = L[M']$, we have then $\pi^N_W(t_N) = [\,] \neq [\pi^{N'}_{M'}(n')] = \pi^N_W(t'_N)$. This implies $\pi^N_W(t_N) = \pi^N_W(t'_N)$ if and only if $W = \lambda$. Suppose $\mathcal{Y} \neq \emptyset$ and $X', Y' \in \mathcal{Y}$ are reconcilable. It follows that $L[X'], L[Y'] \in \mathcal{X}$ are also reconcilable. Consequently, $L[X' \sqcup Y'] = L[X'] \sqcup L[Y'] \in \mathcal{X}$ by assumption, and $X' \sqcup Y' \in \mathcal{Y}$. The hypothesis tells us that there are $t_{N'}, t'_{N'} \in dom(N')$ with $\pi^{N'}_{W'}(t_{N'}) = \pi^{N'}_{W'}(t'_{N'})$ if and only if $W' \in \mathcal{Y}$. We define $t_N = [t_{N'}]$, $t'_N = [t'_{N'}] \in dom(N)$. First, $\pi^N_\lambda(t_N) = \pi^N_\lambda(t'_N)$ holds, and $\lambda \in \mathcal{X}$. For $\lambda \neq W \in Sub(N)$, say $W = L[W']$, we obtain

$$\pi^N_W(t_N) = [\pi^{N'}_{W'}(t_{N'})] = [\pi^{N'}_{W'}(t'_{N'})] = \pi^N_W(t'_N) \quad \text{iff} \quad W' \in \mathcal{Y} \quad \text{iff} \quad W \in \mathcal{X}. \qquad \square$$

The remaining cases of set- and multiset-valued attributes are covered by Lemmas 25 and 29, respectively.

3.2.3. Technical lemmata

We use this section to give some technical definitions and prove some technical results.

Definition 22. Let $N \in \mathcal{N}A$. The identifying term $\tau_N(X)$ of $X \in Sub(N)$ is inductively defined as follows:

- $\tau_\lambda(\lambda) = ok$,
- $\tau_A(\lambda) = a$, $\tau_A(A) = a'$ with $a, a' \in dom(A)$ and $a \neq a'$ for $A \in \mathcal{U}$,
- $\tau_{L(N_1,\ldots,N_k)}(L(M_1, \ldots, M_k)) = (\tau_{N_1}(M_1), \ldots, \tau_{N_k}(M_k))$,
- $\tau_{L\{N\}}(L\{M\}) = \{\tau_N(M)\}$ and $\tau_{L\{N\}}(\lambda) = \emptyset$,
- $\tau_{L\langle N\rangle}(L\langle M\rangle) = \langle\tau_N(M)\rangle$ and $\tau_{L\langle N\rangle}(\lambda) = \langle\,\rangle$,
- $\tau_{L[N]}(L[M]) = [\tau_N(M)]$ and $\tau_{L[N]}(\lambda) = [\,]$.

Fig. 3 shows the subattributes X of $K\{L(A, M[N(B, C)])\}$ together with their identifying terms.

We establish some results on the projection of identifying terms. If the projection of Y's identifying term on X is the same as the projection of X's identifying term on X, then is X necessarily a subattribute of Y.

Lemma 23. *Let $N \in \mathcal{N}A$ and $X, Y \in Sub(N)$. Then $\pi^N_X(\tau_N(Y)) = \pi^N_X(\tau_N(X))$ implies $X \leqslant Y$.*

Proof. We will show the contraposition by induction on N. From $X \nleqslant Y$ follows $X \neq \lambda$.

Let $N = A$ be flat attribute. For $X \nleqslant Y$ it remains to consider the case where $X = A$ and $Y = \lambda$. Then $\pi^N_X(\tau_N(Y)) = \tau_A(\lambda) = a$ and $\pi^N_X(\tau_N(X)) = \tau_A(A) = a'$. This shows $\pi^N_X(\tau_N(Y)) \neq \pi^N_X(\tau_N(X))$.

Let $N = L(N_1, \ldots, N_k)$, $X = L(X_1, \ldots, X_k)$ and $Y = L(Y_1, \ldots, Y_k)$. From $X \not\leqslant Y$ follows $X_i \not\leqslant Y_i$ for some i with $1 \leqslant i \leqslant k$. We conclude that $\pi^{N_i}_{X_i}(\tau_{N_i}(Y_i)) \neq \pi^{N_i}_{X_i}(\tau_{N_i}(X_i))$ holds by hypothesis. However, since $\pi^N_X(\tau_N(Y)) = \pi^N_X(\tau_N(X))$ is equivalent to the fact that $\pi^{N_j}_{X_j}(\tau_{N_j}(Y_j)) = \pi^{N_j}_{X_j}(\tau_{N_j}(X_j))$ holds for all $j = 1, \ldots, k$ the statement of the lemma follows for this case.

Let $N = L\{N'\}$. Then we distinguish between two cases. First, let $Y = \lambda$ and $X = L\{X'\}$. Then we have

$$\pi^N_X(\tau_N(X)) = \pi^{L\{N'\}}_{L\{X'\}}(\tau_{L\{N'\}}(L\{X'\})) = \pi^{L\{N'\}}_{L\{X'\}}(\{\tau_{N'}(X')\}) = \{\pi^{N'}_{X'}(\tau_{N'}(X'))\},$$

but

$$\pi^N_X(\tau_N(Y)) = \pi^{L\{N'\}}_{L\{X'\}}(\tau_{L\{N'\}}(\lambda)) = \pi^{L\{N'\}}_{L\{X'\}}(\emptyset) = \emptyset.$$

It remains the case where $Y = L\{Y'\}$ and $X = L\{X'\}$. From $X' \not\leqslant Y'$ follows $\pi^{N'}_{X'}(\tau_{N'}(Y')) \neq \pi^{N'}_{X'}(\tau_{N'}(X'))$ by hypothesis. It follows that

$$\pi^N_X(\tau_N(Y)) = \{\pi^{N'}_{X'}(\tau_{N'}(Y'))\} \neq \{\pi^{N'}_{X'}(\tau_{N'}(X'))\} = \pi^N_X(\tau_N(X)).$$

+ The proof for the remaining cases of multiset- and list-valued attributes are completely analogous to the case of set-valued attributes. The analogy is due to Definition 22 and the replacement of one-element sets (the empty set) by one-element multisets (the empty multiset) and one-element lists (the empty list), respectively. □

The projection of X's identifying term on Y is the projection of $X \sqcap Y$'s identifying term on Y.

Lemma 24. *Let $N \in \mathcal{N}A$, and $X, Y \in Sub(N)$. Then we have $\pi^N_Y(\tau_N(X)) = \pi^N_Y(\tau_N(X \sqcap Y))$.*

Proof. If $Y = \lambda$, then there is nothing to show. If $N = Y$, then $X \sqcap Y = X \sqcap N = X$. If $X \leqslant Y$, then $X \sqcap Y = X$. In both cases the lemma is obviously true.

We proceed by induction on N. The cases where $N = \lambda$ or N is a flat attribute follow from the considerations above. Suppose $N = L(N_1, \ldots, N_k)$, $Y = L(Y_1, \ldots, Y_k)$ and $X = L(X_1, \ldots, X_k)$. We compute

$$\begin{aligned}\pi^N_Y(\tau_N(X)) &= (\pi^{N_1}_{Y_1}(\tau_{N_1}(X_1)), \ldots, \pi^{N_k}_{Y_k}(\tau_{N_k}(X_k)))\\ &= (\pi^{N_1}_{Y_1}(\tau_{N_1}(X_1 \sqcap Y_1), \ldots, \pi^{N_k}_{Y_k}(\tau_{N_k}(X_k \sqcap Y_k))\\ &= \pi^N_Y(\tau_N(X \sqcap Y)).\end{aligned}$$

Let $N = L\{N'\}$, $Y = L\{Y'\}$ and $X = L\{X'\}$. It follows

$$\begin{aligned}\pi^N_Y(\tau_N(X \sqcap Y)) &= \pi^{L\{N'\}}_{L\{Y'\}}(\tau_{L\{N'\}}(L\{X'\} \sqcap L\{Y'\}))\\ &= \pi^{L\{N'\}}_{L\{Y'\}}(\tau_{L\{N'\}}(L\{X' \sqcap Y'\}))\\ &= \pi^{L\{N'\}}_{L\{Y'\}}(\{\tau_{N'}(X' \sqcap Y')\})\\ &= \{\pi^{N'}_{Y'}(\tau_{N'}(X' \sqcap Y'))\}\\ &= \{\pi^{N'}_{Y'}(\tau_{N'}(X'))\}\\ &= \pi^{L\{N'\}}_{L\{Y'\}}(\{\tau_{N'}(X')\})\\ &= \pi^{L\{N'\}}_{L\{Y'\}}(\tau_{L\{N'\}}(L\{X'\}))\\ &= \pi^N_Y(\tau_N(X)).\end{aligned}$$

The proof for the remaining cases of multiset- and list-valued attributes are completely analogous to the case of set-valued attributes. The analogy is due to Definition 22 and the replacement of one-element sets (the empty set) by one-element multisets (the empty multiset) and one-element lists (the empty list), respectively. □

3.2.4. The case of sets

The construction in the case of set-valued attributes $L\{P\}$ is based on the following idea. Given some ideal $\mathcal{Y}$ of subattributes of P, one element contains exactly the identifying terms of subattributes in $\mathcal{Y}$ while the other element contains the identifying terms of all subattributes of P.

Lemma 25. *Let $N = L\{P\} \in \mathcal{N}A$, and $\emptyset \neq \mathcal{X} \subseteq Sub(N)$ an ideal with respect to $\leqslant$. Then there are $t_N, t'_N \in dom(N)$ with $\pi^N_W(t_N) = \pi^N_W(t'_N)$ if and only if $W \in \mathcal{X}$.*

Proof. Since $\mathcal{X} \neq \emptyset$ is an ideal we have $\lambda \in \mathcal{X}$. Let $\mathcal{X} = \{L\{X\} : X \in \mathcal{Y}\} \cup \{\lambda\}$ for some $\mathcal{Y} \subseteq Sub(P)$. Let $t_N = \{\tau_P(X) : X \leqslant P\}$ and $t'_N = \{\tau_P(X) : X \in \mathcal{Y}\}$. For $W = \lambda$ we obviously have $\pi^N_\lambda(t_N) = ok = \pi^N_\lambda(t'_N)$. Let now be $W = L\{V\}$. We need to show that

$$\{\pi^P_V(\tau_P(X)) : X \leqslant P\} = \{\pi^P_V(\tau_P(X)) : X \in \mathcal{Y}\} \quad \text{if and only if} \quad V \in \mathcal{Y}$$

holds. It is always true that $\{\pi^P_V(\tau_P(X)) : X \in \mathcal{Y}\} \subseteq \{\pi^P_V(\tau_P(X)) : X \leqslant P\}$ holds since $\mathcal{Y} \subseteq Sub(P)$.

We show first that $V \in \mathcal{Y}$ implies $\{\pi^P_V(\tau_P(X)) : X \leqslant P\} \subseteq \{\pi^P_V(\tau_P(X)) : X \in \mathcal{Y}\}$. Let $V \in \mathcal{Y}$. We show that for all $X \leqslant P$ there is some $Y \in \mathcal{Y}$ with $\pi^P_V(\tau_P(X)) = \pi^P_V(\tau_P(Y))$. If $X \in \mathcal{Y}$, then obviously take $Y = X$. If $X \notin \mathcal{Y}$, then take $Y = X \sqcap V$. We conclude $\pi^P_V(\tau_P(X)) = \pi^P_V(\tau_P(Y))$ by Lemma 24. Certainly, $Y \in \mathcal{Y}$ since $\mathcal{Y}$ is an $\leqslant$-ideal.

It remains to show that $\{\pi^P_V(\tau_P(X)) : X \in \mathcal{Y}\} \subset \{\pi^P_V(\tau_P(X)) : X \leqslant P\}$, if $V \notin \mathcal{Y}$. Let $V \notin \mathcal{Y}$. Since $\mathcal{Y}$ is an ideal it follows that all $X \leqslant P$ with $V \leqslant X$ also satisfy $X \notin \mathcal{Y}$. Hence, $\tau_P(X) \in t_N$, but $\tau_P(X) \notin t'_N$ for all X with $V \leqslant X \leqslant P$. Suppose there was some $X \in \mathcal{Y}$ with $\pi^P_V(\tau_P(X)) = \pi^P_V(\tau_P(V))$. Using Lemma 23 we infer $V \leqslant X$ and therefore $\tau_P(X) \notin t'_N$. This is a contradiction since $\tau_P(X) \in t'_N$ for all $X \in \mathcal{Y}$ holds. Consequently, $\pi^P_V(\tau_P(X)) \neq \pi^P_V(\tau_P(V))$ for all $X \notin \mathcal{Y}$. We conclude that $\pi^P_V(\tau_P(V)) \in \{\pi^P_V(\tau_P(X)) : X \leqslant P\}$ and $\pi^P_V(\tau_P(V)) \notin \{\pi^P_V(\tau_P(X)) : X \in \mathcal{Y}\}$. This concludes the proof. □

Example 26. Consider the nested attribute $N = K\{L(A, M[O(B, C)])\}$ together with the FDs $K\{L(A)\} \rightarrow K\{L(M[O(B)])\}$ and $K\{L(A)\} \rightarrow K\{L(M[O(C)])\}$. The closure $\mathcal{X}^+$ of $\mathcal{X} = K\{L(A)\}$ with respect to these FDs is illustrated in Fig. 4.

We generate two elements t_N, t'_N which coincide exactly on the elements of $\mathcal{X}^+$. Following the proof of Lemma 25, $t_N = \{\tau_{L(A,M[O(B,C)])}(X) : X \leqslant L(A, M[O(B, C)])\}$ is

$$\{(a', [(b', c')]); (a, [(b', c')]); (a', [(b', c)]); (a', [(b, c')]); (a, [(b', c)]); (a, [(b, c')]); (a', [(b, c)]); (a, [(b, c)]); (a', [\,]); (a, [\,])\}$$

and $t'_N = \{\tau_{L(A,M[O(B,C)])}(Y) : Y \in \mathcal{Y}\}$ is

$$\{(a, [(b', c)]); (a, [(b, c')]); (a, [(b, c)]); (a', [\,]); (a, [\,])\}.$$

The projections $\pi^N_W(t)$ and $\pi^N_W(t')$ for $W \in Sub(N)$ are:

W	$\pi^N_W(t_N)$	$\pi^N_W(t'_N)$
$K\{L(M[O(B)])\}$	$\{(ok, [(b', ok)]); (ok, [(b, ok)]); (ok, [\,])\}$	
$K\{L(M[O(C)])\}$	$\{(ok, [(ok, c')]); (ok, [(ok, c)]); (ok, [\,])\}$	
$K\{L(A)\}$	$\{(a', ok); (a, ok)\}$	
$K\{L(M[O(B, C)])\}$	$\{(ok, [(b, c)]); (ok, [(b', c)]), (ok, [(b, c')]); (\mathbf{ok}, [(\mathbf{b}', \mathbf{c}')]); (ok, [\,])\}$	$\{(ok, [(b, c)]); (ok, [(b', c)]); (ok, [(b, c')]); (ok, [\,])\}$
$K\{L(A, M[\lambda])\}$	$\{(a, [(ok, ok)]); (a, [\,]); (\mathbf{a}', [(\mathbf{ok}, \mathbf{ok})]); (a', [\,])\}$	$\{(a, [(ok, ok)]); (a, [\,]); (a', [\,])\}$

Indeed, t_N and t'_N coincide on all maximal elements of $\mathcal{X}^+$, and therefore on all elements of $\mathcal{X}^+$. Furthermore, t_N and t'_N deviate on all minimal attributes of $Sub(N)$ which are not in $\mathcal{X}^+$.

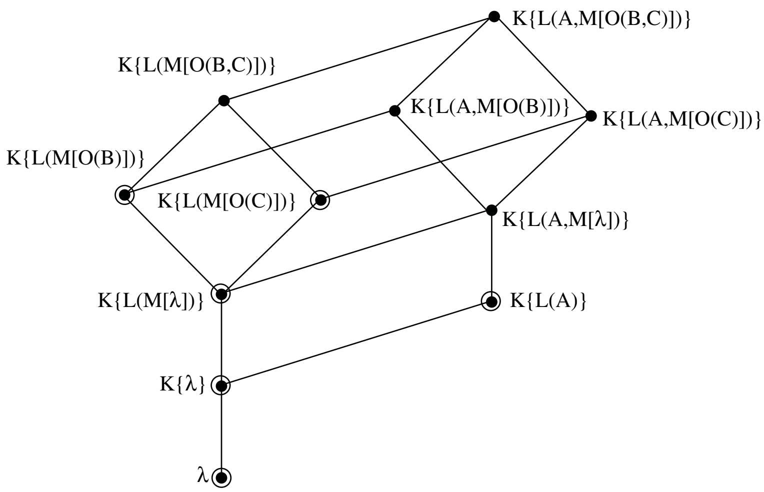

Fig. 4. The closure $\mathcal{X}^+$ of $\mathcal{X} = K\{L(A)\}$.

3.2.5. The case of multisets

The strategy used for set-valued attributes does not work for multiset-valued attributes since multiple occurrences of projections do not vanish in a multiset. At this point it helps to look deeper into the structure of nested attributes. In fact, the relativised subalgebra $(Sub(X), \leqslant, \sqcup, \sqcap, \dot{-}, X)$ with respect to $X \sqcap X_1 \sqcap \cdots \sqcap X_k$ is Boolean where $X_1, \ldots, X_k$ are the $\leqslant$-maximal subattributes of $X \in Sub(N)$.

Let $X, Y \in Sub(N)$ with $X \leqslant Y$. Then $[X, Y] = \{Z \in Sub(N) : X \leqslant Z \leqslant Y\}$ is called an *interval* of $Sub(N)$ [3,34].

Lemma 27. *Let $N \in \mathcal{N}A$ and $X \in Sub(N)$. Let $\{X_1, \ldots, X_k\}$ be the set of all $\leqslant$-maximal proper subattributes of X. Then $([0_X, X], \leqslant, \sqcap, \sqcup, \overline{(\cdot)}, 0_X, X)$ forms a Boolean algebra where $0_X = X \sqcap X_1 \sqcap \cdots \sqcap X_k$ and $\overline{Y} = (X \dot{-} Y) \sqcup 0_X$ for all $Y \in [0_X, X]$.*

Proof. The order $\leqslant$, meet $\sqcap$ and join $\sqcup$ in $([0_X, X], \leqslant, \sqcap, \sqcup)$ are the respective restrictions of order, meet and join from $(Sub(N), \leqslant, \sqcap, \sqcup)$ to $[0_X, X]$. $[0_X, X]$ is closed under meet $\sqcap$, join $\sqcup$ and complement $\overline{(\cdot)}$. It remains to show that $\overline{Y} = (X \dot{-} Y) \sqcup 0_X$ defines indeed the complement of $Y \in [0_X, X]$.

We show that $Y \sqcap (X \dot{-} Y) \leqslant 0_X$ holds for all $Y \in [0_X, X]$. If $Y = X$, then $Y \sqcap (X \dot{-} Y) = \lambda_N \leqslant 0_X$. If $Y = 0_X$, then $Y \sqcap (X \dot{-} Y) = 0_X \leqslant 0_X$. For every other $Y \in [0_X, X]$ we have then $Y = X_{i_1} \sqcap \cdots \sqcap X_{i_n}$ where $\{1, \ldots, k\}$ is the disjoint union of the two non-empty sets $\{i_1, \ldots, i_n\}$ and $\{j_1, \ldots, j_m\}$. Since $(X_{i_1} \sqcap \cdots \sqcap X_{i_n}) \sqcup (X_{j_1} \sqcap \cdots \sqcap X_{j_m}) = (X_{i_1} \sqcup X_{j_1}) \sqcap \cdots \sqcap (X_{i_n} \sqcup X_{j_m}) = X$ holds (the join of two different maximal proper subattributes of X is always X) we have $X \dot{-} Y \leqslant X_{j_1} \sqcap \cdots \sqcap X_{j_m}$. We conclude that $Y \sqcap (X \dot{-} Y) \leqslant X_{i_1} \sqcap \cdots \sqcap X_{i_n} \sqcap X_{j_1} \sqcap \cdots \sqcap X_{j_m} = X_1 \sqcap \cdots \sqcap X_k = X \sqcap X_1 \sqcap \cdots \sqcap X_k = 0_X$.

It follows then that $Y \sqcup \overline{Y} = Y \sqcup (X \dot{-} Y) \sqcup 0_X = X \sqcup Y \sqcup 0_X = X$, and $Y \sqcap \overline{Y} = Y \sqcap ((X \dot{-} Y) \sqcup 0_X) = (Y \sqcap (X \dot{-} Y)) \sqcup (Y \sqcap 0_X) \leqslant 0_X \sqcup 0_X = 0_X$. This completes the proof. □

We are going to prove the existence of two elements which deviate in their projections on exactly all elements of a principal filter, i.e., on all elements in the shaded area of the left picture in Fig. 5. Recall that a filter [3,34] of some poset $(S, \leqslant)$ is a subset $\mathcal{F} \subseteq S$ that is closed upwards with respect to $\leqslant$, i.e., if $X \in \mathcal{F}$ and $X \leqslant Y$, then $Y \in \mathcal{F}$ as well. A principal filter of $(S, \leqslant)$ is a filter of $(S, \leqslant)$ that is generated from a single element of S.

The idea is to use a bijection between the intervals $[0_Y, Y \sqcap U]$ and $[\overline{Y \sqcap U}, Y]$. The meet of Y and some $\leqslant$-maximal subattribute U of Y that is not in the principal filter of Y, however, is always the complement of some atom. This is illustrated in the right-hand picture of Fig. 5. One multiset contains the identifying terms of all attributes from the even levels of $([0_Y, Y], \leqslant)$, the other multiset contains the identifying terms of all attributes from the odd levels of $([0_Y, Y], \leqslant)$. The kth level of $([0_Y, Y], \leqslant)$ is defined as the set of all elements in $[0_Y, Y]$ that have distance k to 0_Y in the Hasse diagram of $([0_Y, Y], \leqslant)$, see also [3,34].

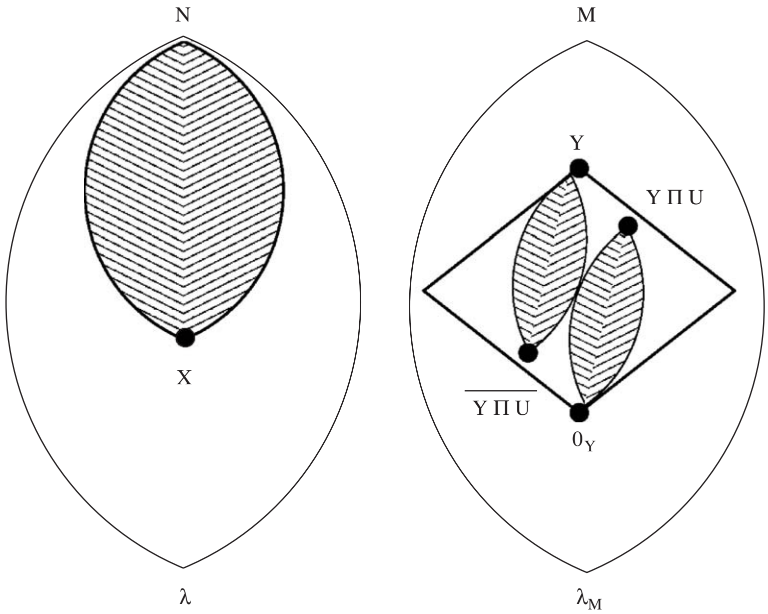

Fig. 5. Illustration of Lemma 28.

Lemma 28. *Let $N = L\langle M\rangle \in \mathcal{N}A$ and $\lambda \neq X = L\langle Y\rangle \leqslant N$. Then there are $t_1, t_2 \in dom(N)$ with $\pi_W^N(t_1) \neq \pi_W^N(t_2)$ for $W \in Sub(N)$ if and only if $X \leqslant W$.*

Proof. Let $([0_Y, Y], \leqslant, \sqcap, \sqcup, \overline{(\cdot)}, 0_Y, Y)$ be the Boolean algebra according to Lemma 27 where $[0_Y, Y]$ contains 2^k elements. Let $\mathcal{L}_i$ denote the ith level of $([0_Y, Y], \leqslant)$ for $i = 0, \ldots, k$. Then we define $t_1 = \langle \tau_M(Z) : Z \in \mathcal{L}_i, i \text{ even}\rangle$ and $t_2 = \langle \tau_M(Z) : Z \in \mathcal{L}_i, i \text{ odd}\rangle$. Note that $t_2 = \langle\ \rangle$, if $k = 0$.

First, it follows that $\pi_Y^M(\tau_M(Y))$ is an element of either $\pi_X^N(t_1)$ or $\pi_X^N(t_2)$. If $\pi_Y^M(\tau_M(Y)) = \pi_Y^M(\tau_M(Z))$ held for some $Z \leqslant M$, then $Y \leqslant Z$ by Lemma 23. The elements t_1 and t_2, however, have only identifying terms of subattributes $Z \leqslant Y$ as members. We conclude that $\pi_Y^M(\tau_M(Y)) \neq \pi_Y^M(\tau_M(Z))$ for $Z < Y$. This shows that $\pi_X^N(t_1) \neq \pi_X^N(t_2)$, and therefore also $\pi_W^N(t_1) \neq \pi_W^N(t_2)$ whenever $X \leqslant W$.

It remains to show that $\pi_W^N(t_1) = \pi_W^N(t_2)$ holds whenever $X \nleqslant W$ holds. It is sufficient to show that $\pi_V^N(t_1) = \pi_V^N(t_2)$ holds for all $\leqslant$-maximal subattributes $V \in Sub(N)$ with $X \nleqslant V$. This is obvious if $V = \lambda$. Let therefore be $V = L\langle U\rangle$ where U is a $\leqslant$-maximal subattribute $U \in Sub(M)$ with $Y \nleqslant U$.

We show first that $Y \sqcap U$ is always a $\leqslant$-maximal proper subattribute of Y. Suppose there is some Z with $Y \sqcap U < Z < Y$. If $U = Z \sqcup U$, then

$$U \sqcap Y = (Z \sqcup U) \sqcap Y = (Z \sqcap Y) \sqcup (U \sqcap Y) = Z \sqcup (U \sqcap Y) = Z$$

and this contradicts $U \sqcap Y < Z$. This shows $U < Z \sqcup U$. If $Y \leqslant Z \sqcup U$, then $Y \dot{-} Z \leqslant U \sqcap Y \leqslant Z$. This means $Y \leqslant Z$ which gives the contradiction $Z < Y \leqslant Z$. We conclude that $U < Z \sqcup U$ and $Y \nleqslant Z \sqcup U$. This contradicts the $\leqslant$-maximality of U with $Y \nleqslant U$ and shows that $Z = Y \sqcap U$ or $Z = Y$, i.e., $Y \sqcap U$ is indeed a $\leqslant$-maximal proper subattribute of Y. This implies that $Y \sqcap U$ is always the complement of an atom of $([0_Y, Y], \leqslant)$.

Let $[0_Y, Y \sqcap U]$, $[\overline{Y \sqcap U}, Y]$ denote the intervals between 0_Y and $Y \sqcap U$, and $\overline{Y \sqcap U}$ and Y, respectively. The mapping $Z \mapsto Z \sqcup \overline{Y \sqcap U}$ from $[0_Y, Y \sqcap U]$ to $[\overline{Y \sqcap U}, Y]$ is bijective with inverse $Z \mapsto Z \sqcap (Y \sqcap U)$. Since $\overline{Y \sqcap U}$ is an atom we have $\tau_M(Z \sqcup \overline{Y \sqcap U}) \in t_2$ whenever $\tau_M(Z) \in t_1$, and vice versa. The situation is illustrated in the right picture of Fig. 5.

It is now sufficient to show that $\pi_U^M(\tau_M(Z)) = \pi_U^M(\tau_M(Z \sqcup \overline{Y \sqcap U}))$ for $Z \in [0_Y, Y \sqcap U]$. We have

$$\begin{aligned}\pi_U^M(\tau_M(Z)) &= \pi_U^M(\tau_M(Z \sqcup 0_Y))\\ &= \pi_U^M(\tau_M((Z \sqcap U) \sqcup (\overline{Y \sqcap U} \sqcap Y \sqcap U)))\end{aligned}$$

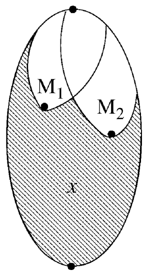

Fig. 6. Illustration of Lemma 29.

$$= \pi_U^M(\tau_M((Z \sqcap U) \sqcup (\overline{Y \sqcap U} \sqcap U)))$$
$$= \pi_U^M(\tau_M((Z \sqcup \overline{Y \sqcap U}) \sqcap U))$$
$$= \pi_U^M(\tau_M(Z \sqcup \overline{Y \sqcap U})),$$

where the last equation follows from Lemma 24. □

For the general construction we pick all $\leqslant$-minimal subattributes M_i that are not in the ideal $\mathcal{X}$ and form the union over all multisets given by the previous lemma on all generated principal filters. This is illustrated by Fig. 6.

Lemma 29. *Let $N = L\langle P\rangle \in \mathcal{N}A$, and $\emptyset \neq \mathcal{X} \subseteq Sub(N)$ an ideal with respect to $\leqslant$. Then there are $t_N, t'_N \in dom(N)$ with $\pi_W^N(t_N) = \pi_W^N(t'_N)$ if and only if $W \in \mathcal{X}$.*

Proof. Let $\{M_1, \ldots, M_n\} \subseteq Sub(N)$ be the set of all $\leqslant$-minimal subattributes of N with $M_i \notin \mathcal{X}$. Since $\lambda \in \mathcal{X}$ holds it follows that $M_i \neq \lambda$ for all $i = 1, \ldots, n$. According to Lemma 28, and for all $i = 1, \ldots, n$, there are $t_{M_i}, t'_{M_i} \in dom(N)$ with $\pi_Z^N(t_{M_i}) \neq \pi_Z^N(t'_{M_i})$ if and only if $M_i \leqslant Z$. Define $t_N = \bigcup_{i=1}^n t_{M_i}$ and $t'_N = \bigcup_{i=1}^n t'_{M_i}$, where the union is taken over multisets. If $W \in \mathcal{X}$ holds, then $M_i \nleqslant W$ for all $i = 1, \ldots, n$ and, consequently $\pi_W^N(t_{M_i}) = \pi_W^N(t'_{M_i})$ holds for all $i = 1, \ldots, n$ as well. This implies $\pi_W^N(t_N) = \pi_W^N(t'_N)$. If $W \notin \mathcal{X}$ holds, then there is some j with $1 \leqslant j \leqslant n$ such that $M_j \leqslant W$ holds. The element $\pi_W^N(\tau_N(M_j))$, however, is member of exactly one of $\pi_W^N(t_N)$, $\pi_W^N(t'_N)$ by the construction. This implies $\pi_W^N(t_N) \neq \pi_W^N(t'_N)$. Consequently, $\pi_W^N(t_N) = \pi_W^N(t'_N)$ if and only if $W \in \mathcal{X}$. □

Example 30. We will illustrate the construction for multisets. Consider the nested attribute $N = L\langle M\rangle$ with $M = K(J[A], O\{P(B, Q\{C\})\})$. The structure of $(Sub(M), \leqslant)$ is illustrated in Fig. 7 where labels have been omitted.

Let $\mathcal{X} = \{L\langle X\rangle : X \in \mathcal{Y}\}$, where $\mathcal{Y}$ is the ideal that consists of all subattributes of M which are circled in Fig. 7. The $\leqslant$-minimal subattributes $V \in Sub(N)$ with $V \notin \mathcal{X}$ are $V_1 = L\langle K(J[\lambda], O\{P(B, Q\{\lambda\})\})\rangle$ and $V_2 = L\langle K(J[A], O\{P(\lambda, \lambda)\})\rangle$. The structures of $([K(\lambda, O\{P(\lambda, \lambda)\}), K(J[\lambda], O\{P(B, Q\{\lambda\})\})], \leqslant)$ and $([K(J[\lambda], \lambda), K(J[A], O\{P(\lambda, \lambda)\})], \leqslant)$ are illustrated in Fig. 8.

According to Lemma 28 the following elements are chosen:

$$t'_1 = \langle([\,], \{(b, \emptyset)\}); ([\,], \{(b', \{c\})\}); ([a], \{(b', \emptyset)\}); ([a], \{(b, \{c\})\})\rangle,$$
$$t'_2 = \langle([\,], \{(b', \emptyset)\}); ([\,], \{(b, \{c\})\}); ([a], \{(b, \emptyset)\}); ([a], \{(b', \{c\})\})\rangle,$$
$$t''_1 = \langle([a], \emptyset); ([a'], \{(b, \emptyset)\})\rangle,$$
$$t''_2 = \langle([a], \{(b, \emptyset)\}); ([a'], \emptyset)\rangle.$$

Finally, and according to Lemma 29 one chooses

$$t_N = t'_1 \cup t''_1 = \langle([\,], \{(b, \emptyset)\}); ([\,], \{(b', \{c\})\}); ([a], \{(b', \emptyset)\}); ([a], \{(b, \{c\})\}); ([a], \emptyset); ([a'], \{(b, \emptyset)\})\rangle,$$
$$t'_N = t'_2 \cup t''_2 = \langle([\,], \{(b', \emptyset)\}); ([\,], \{(b, \{c\})\}); ([\mathbf{a}], \{(\mathbf{b}, \emptyset)\}); ([a], \{(b', \{c\})\}); ([\mathbf{a}], \{(\mathbf{b}, \emptyset)\}); ([a'], \emptyset)\rangle.$$

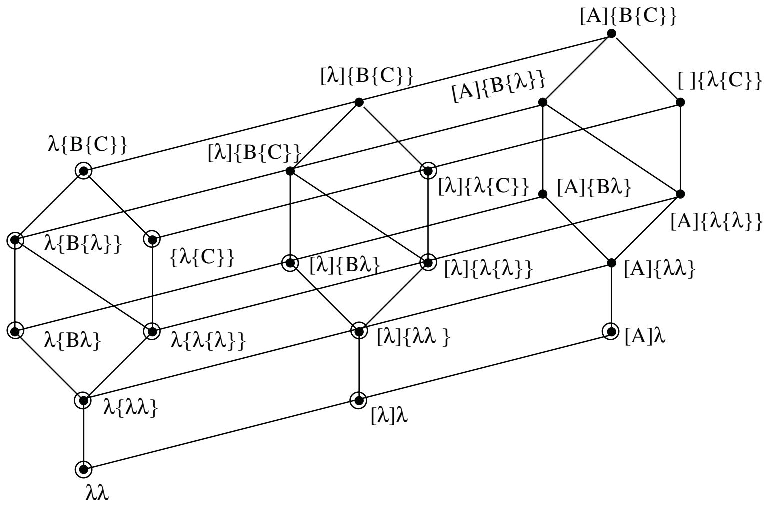

Fig. 7. The structure of $M = K(J[A], O\{P(B, Q\{C\})\})$.

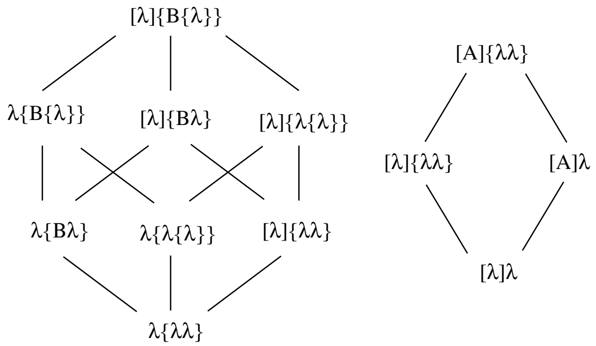

Fig. 8. The structure of Subalgebras in Example 30.

One can verify then that $\pi^N_W(t_N) = \pi^N_W(t'_N)$ for all $\leqslant$-maximal $W \in \mathcal{X}$, i.e., $W \in \{L\langle K(\lambda, O\{P(B, Q\{C\})\})\rangle$, $L\langle K(J[\lambda], O\{P(B, \lambda)\})\rangle$, $L\langle K(J[\lambda], O\{P(\lambda, Q\{\lambda\})\})\rangle$, $L\langle K(J[A], \lambda)\rangle\}$. Furthermore, $\pi^N_{V_1}(t_N) \neq \pi^N_{V_1}(t'_N)$ and $\pi^N_{V_2}(t_N) \neq \pi^N_{V_2}(t'_N)$.

3.3. A note on reconcilability

We demonstrate that reconcilability of X and Y is an exact condition for the soundness of the restricted join axiom $\dfrac{}{\{X, Y\} \to \{X \sqcup_N Y\}}$. This means that one cannot find a weaker sufficient condition for that rule to hold. Proposition 18 already implies that reconcilability is a sufficient condition. If X and Y are not reconcilable, then we show that there is some instance r with $\not\models_r \{X, Y\} \to \{X \sqcup Y\}$. It is then sufficient to find an ideal $\mathcal{Y}$ satisfying the properties of Lemma 21 and where $X, Y \in \mathcal{Y}$, but $X \sqcup Y \notin \mathcal{Y}$. This guarantees the existence of t_N, t'_N with $\pi^N_W(t_N) = \pi^N_W(t'_N)$ if and only if $W \in \mathcal{Y}$. The desired r is then $\{t_N, t'_N\}$.

Lemma 31. *Let $N \in \mathcal{N}A$ and $X, Y \in Sub(N)$. Then $\mathcal{Y} = \{U \sqcup V : U \leqslant X, V \leqslant Y, U$ and V are reconcilable$\}$ is a non-empty ideal with respect to $\leqslant$ and for all $S, T \in \mathcal{Y}$ that are reconcilable follows $S \sqcup T \in \mathcal{Y}$.*

Proof. $\mathcal{Y}$ is non-empty as $\lambda \in \mathcal{Y}$ holds. We show that $\mathcal{Y}$ is an ideal with respect to $\leqslant$. Let $S \in \mathcal{Y}$, i.e., $S = U \sqcup V$ with $U \leqslant X$, $V \leqslant Y$ and U, V are reconcilable. Let $T \leqslant S$. Then $T = S \sqcap T = (U \sqcup V) \sqcap T = (U \sqcap T) \sqcup (V \sqcap T)$

where $U \sqcap T \leqslant U \leqslant X$ and $V \sqcap T \leqslant V \leqslant Y$ holds. We show that $U \sqcap T$, $V \sqcap T$ are reconcilable, and conclude that $T \in \mathcal{Y}$, too. We proceed by induction on reconcilable nested attributes. If $U \leqslant V$, then $U \sqcap T \leqslant V \sqcap T$. Similarly, if $V \leqslant U$, then $V \sqcap T \leqslant U \sqcap T$. If $T = \lambda$, then $U \sqcap T = V \sqcap T$. Let $N = L(N_1, \ldots, N_k)$, $U = L(U_1, \ldots, U_k)$, $V = L(V_1, \ldots, V_k)$ and $T = L(T_1, \ldots, T_k)$. Since U, V are reconcilable it follows that U_i, V_i are reconcilable for all $i = 1, \ldots, k$. Consequently, $U_i \sqcap V_i$ and $V_i \sqcap T_i$ are also reconcilable for $i = 1, \ldots, k$. The reconcilability of $U \sqcap T$ and $V \sqcap T$ follows from the fact that $U \sqcap T = L(U_1 \sqcap T_1, \ldots, U_k \sqcap T_k)$ and $V \sqcap T = L(V_1 \sqcap T_1, \ldots, V_k \sqcap T_k)$. Let $N = L[N']$, $U = L[U']$, $V = L[V']$ and $T = L[T']$. Then U', V' are reconcilable by definition, and $U' \sqcap T'$, $V' \sqcap T'$ are reconcilable as well. Since $U \sqcap T = L[U' \sqcap T']$ and $V \sqcap T = L[V' \sqcap T']$ it is proven that $U \sqcap T$ and $V \sqcap T$ are indeed reconcilable.

It remains to show that $\mathcal{Y}$ is closed under the join of reconcilable elements. Let $S, T \in \mathcal{Y}$ be reconcilable. We proceed again by induction on the definition of reconcilable nested attributes in order to show that $S \sqcup T \in \mathcal{Y}$ holds as well. Note that this is true, if $X = \lambda$ or $Y = \lambda$. If $S \leqslant T$, then $S \sqcup T = T \in \mathcal{Y}$, and if $T \leqslant S$, then $S \sqcup T = S \in \mathcal{Y}$. Let $N = L(N_1, \ldots, N_k)$, $X = L(X_1, \ldots, X_k)$, $Y = L(Y_1, \ldots, Y_k)$. It follows that $\mathcal{Y} = \{L(M_1, \ldots, M_k) : M_i \in \mathcal{Y}_i\}$ where $\mathcal{Y}_i = \{U_i \sqcup V_i : U_i \leqslant X_i, V_i \leqslant Y_i \text{ and } U_i, V_i \text{ are reconcilable}\}$ is a non-empty ideal for every $i = 1, \ldots, k$. Let $S, T \in \mathcal{Y}$ be reconcilable. Then $S = L(S_1, \ldots, S_k)$, $T = L(T_1, \ldots, T_k)$ with $S_i, T_i \in \mathcal{Y}_i$ for $i = 1, \ldots, k$. Furthermore, S_i, T_i are reconcilable. We know that $S_i \sqcup T_i \in \mathcal{Y}_i$ holds for every $i = 1, \ldots, k$, and therefore $S \sqcup T = L(S_1, \ldots, S_k) \sqcup L(T_1, \ldots, T_k) = L(S_1 \sqcup T_1, \ldots, S_k \sqcup T_k) \in \mathcal{Y}$ which proves this case.

Let $N = L[N']$, $X = L[X']$, $Y = L[Y']$. It follows that $\mathcal{Y} = \{L[M] : M \in \mathcal{Y}'\} \cup \{\lambda\}$ where $\mathcal{Y}' = \{U' \sqcup V' : U' \leqslant X', V' \leqslant Y' \text{ and } U', V' \text{ are reconcilable}\}$ is a non-empty ideal. If $\mathcal{Y}' = \emptyset$, then $\mathcal{Y} = \{\lambda\}$ and $S \sqcup T = \lambda \in \mathcal{Y}$. Let $S, T \in \mathcal{Y}$ be reconcilable, say $S = L[S']$ and $T = L[T']$. Consequently, $S', T' \in \mathcal{Y}'$, and the reconcilability of S', T' follows from the reconcilability of S, T. We know that $S' \sqcup T' \in \mathcal{Y}'$ which means that $S \sqcup T = L[S'] \sqcup L[T'] = L[S' \sqcup T'] \in \mathcal{Y}$ holds. □

4. Minimality

We will investigate whether the generalised Armstrong axioms form a minimal, sound and complete set of inference rules for the implication of FDs in the sense of the following definition.

Definition 32. Let $\mathfrak{R}$ denote some set of inference rules. An inference rule R is independent from $\mathfrak{R}$ if and only if there is a nested attribute N and a set Σ of dependencies on N as well as some dependency σ with $\sigma \notin \Sigma^+_{\mathfrak{R}}$ but $\sigma \in \Sigma^+_{\mathfrak{R} \cup \{R\}}$. A sound and complete set $\mathfrak{R}$ of inference rules is called minimal for the implication of dependencies if and only if every $R \in \mathfrak{R}$ is independent from $\mathfrak{R} - \{R\}$, i.e., there is no $\mathfrak{R}' \subset \mathfrak{R}$ which is complete as well.

We will now show that each of the generalised Armstrong axioms is independent from the rest of the rules.

Lemma 33. *The reflexivity axiom is independent from $\mathfrak{R} = \{$subattribute axiom, extension rule, restricted join axiom, transitivity rule$\}$.*

Proof. Let $N = L\{A\}$, $\Sigma = \emptyset$ and $\sigma = \{\lambda, L\{\lambda\}, L\{A\}\} \to \{\lambda\}$. We present $\Sigma^+_{\mathfrak{R}}$ by the following table where the row names denote the left-hand side $\mathcal{X}$, and the column names denote the right-hand side $\mathcal{Y}$ of an FD $\mathcal{X} \to \mathcal{Y}$. An FD $\mathcal{X} \to \mathcal{Y}$ belongs to $\Sigma^+_{\mathfrak{R}}$ if and only if the entry at row $\mathcal{X}$ and column $\mathcal{Y}$ is a cross ×.

	$\{\lambda\}$	$\{L\{\lambda\}\}$	$\{L\{A\}\}$	$\{\lambda, L\{\lambda\}\}$	$\{\lambda, L\{A\}\}$	$\{L\{\lambda\}, L\{A\}\}$	$\{\lambda, L\{\lambda\}, L\{A\}\}$
$\{\lambda\}$	×						
$\{L\{\lambda\}\}$	×	×		×			
$\{L\{A\}\}$	×	×	×	×	×	×	×
$\{\lambda, L\{\lambda\}\}$	×	×		×			
$\{\lambda, L\{A\}\}$	×	×	×	×	×	×	×
$\{L\{\lambda\}, L\{A\}\}$	×	×	×	×	×	×	×
$\{\lambda, L\{\lambda\}, L\{A\}\}$							

We can see that $\sigma \notin \Sigma^+_{\mathfrak{R}}$. However, as $\{\lambda\} \subseteq \{\lambda, L\{\lambda\}, L\{A\}\}$ we conclude that σ can be inferred from Σ using the reflexivity axiom. □

Lemma 34. *The subattribute axiom is independent from* $\mathfrak{R} = \{$*reflexivity axiom*, *extension rule*, *restricted join axiom*, *transitivity rule*$\}$.

Proof. Let $N = L(A)$, $\Sigma = \emptyset$ and $\sigma = \{L(A)\} \to \{\lambda\}$. The following table represents $\Sigma^+_{\mathfrak{R}}$.

	$\{\lambda\}$	$\{L(A)\}$	$\{\lambda, L(A)\}$
$\{\lambda\}$	×		
$\{L(A)\}$		×	
$\{\lambda, L(A)\}$	×	×	×

We can see that $\sigma \notin \Sigma^+_{\mathfrak{R}}$. However, as $\lambda \leqslant L(A)$ we conclude that σ can be inferred from Σ using the subattribute axiom. □

Lemma 35. *The extension rule is independent from* $\mathfrak{R} = \{$*reflexivity axiom*, *subattribute axiom*, *restricted join axiom*, *transitivity rule*$\}$.

Proof. Let $N = L(A)$, $\Sigma = \emptyset$ and $\sigma = \{L(A)\} \to \{\lambda, L(A)\}$. The following table represents $\Sigma^+_{\mathfrak{R}}$.

	$\{\lambda\}$	$\{L(A)\}$	$\{\lambda, L(A)\}$
$\{\lambda\}$	×		
$\{L(A)\}$	×	×	
$\{\lambda, L(A)\}$	×	×	×

We can see that $\sigma \notin \Sigma^+_{\mathfrak{R}}$. However, as $\{L(A)\} \to \{\lambda\} \in \Sigma^+_{\mathfrak{R}}$ we conclude that σ can be inferred from Σ using the extension rule and $\mathfrak{R}$. □

Lemma 36. *The restricted join axiom is independent from* $\mathfrak{R} = \{$*reflexivity axiom*, *subattribute axiom*, *extension rule*, *transitivity rule*$\}$.

Proof. Let $N = L(A, B)$, $\Sigma = \emptyset$ and $\sigma = \{L(A), L(B)\} \to \{L(A, B)\}$. We compute $\Sigma^+_{\mathfrak{R}}$ by the tables

	$\{\lambda\}$	$\{L(A)\}$	$\{L(B)\}$	$\{L(A, B)\}$	$\{\lambda, L(A)\}$	$\{\lambda, L(B)\}$	$\{\lambda, L(A, B)\}$	$\{L(A), L(B)\}$
$\{\lambda\}$	×							
$\{L(A)\}$	×	×			×			
$\{L(B)\}$	×		×			×		
$\{L(A, B)\}$	×	×	×	×	×	×	×	×
$\{\lambda, L(A)\}$	×	×			×			
$\{\lambda, L(B)\}$	×		×			×		
$\{\lambda, L(A, B)\}$	×	×	×	×	×	×	×	×
$\{L(A), L(B)\}$	×	×	×		×	×		×
$\{L(A), L(A, B)\}$	×	×	×	×	×	×	×	×
$\{L(B), L(A, B)\}$	×	×	×	×	×	×	×	×
$\{\lambda, L(A), L(B)\}$	×	×	×		×	×		×
$\{\lambda, L(A), L(A, B)\}$	×	×	×	×	×	×	×	×
$\{\lambda, L(B), L(A, B)\}$	×	×	×	×	×	×	×	×
$\{L(A), L(B), L(A, B)\}$	×	×	×	×	×	×	×	×
$\{\lambda, L(A), L(B), L(A, B)\}$	×	×	×	×	×	×	×	×

and

	$\{L(A), L(A,B)\}$	$\{L(B), L(A,B)\}$	$\{\lambda, L(A), L(B)\}$	$\{\lambda, L(A), L(A,B)\}$
$\{\lambda\}$				
$\{L(A)\}$				
$\{L(B)\}$				
$\{L(A,B)\}$	×	×	×	×
$\{\lambda, L(A)\}$				
$\{\lambda, L(B)\}$				
$\{\lambda, L(A,B)\}$	×	×	×	×
$\{L(A), L(B)\}$			×	
$\{L(A), L(A,B)\}$	×	×	×	×
$\{L(B), L(A,B)\}$	×	×	×	×
$\{\lambda, L(A), L(B)\}$			×	
$\{\lambda, L(A), L(A,B)\}$	×	×	×	×
$\{\lambda, L(B), L(A,B)\}$	×	×	×	×
$\{L(A), L(B), L(A,B)\}$	×	×	×	×
$\{\lambda, L(A), L(B), L(A,B)\}$	×	×	×	×

and

	$\{\lambda, L(B), L(A,B)\}$	$\{L(A), L(B), L(A,B)\}$	$\{\lambda, L(A), L(B), L(A,B)\}$
$\{\lambda\}$			
$\{L(A)\}$			
$\{L(B)\}$			
$\{L(A,B)\}$	×	×	×
$\{\lambda, L(A)\}$			
$\{\lambda, L(B)\}$			
$\{\lambda, L(A,B)\}$	×	×	×
$\{L(A), L(B)\}$			
$\{L(A), L(A,B)\}$	×	×	×
$\{L(B), L(A,B)\}$	×	×	×
$\{\lambda, L(A), L(B)\}$			
$\{\lambda, L(A), L(A,B)\}$	×	×	×
$\{\lambda, L(B), L(A,B)\}$	×	×	×
$\{L(A), L(B), L(A,B)\}$	×	×	×
$\{\lambda, L(A), L(B), L(A,B)\}$	×	×	×

We can see that $\sigma \notin \Sigma^+_{\mathfrak{R}}$. However, as $L(A)$ and $L(B)$ are reconcilable, we conclude that σ can be inferred from Σ using the restricted join axiom. □

Lemma 37. *The transitivity rule is independent from* $\mathfrak{R} = \{$*reflexivity axiom*, *subattribute axiom*, *extension rule*, *restricted join axiom*$\}$.

Proof. Let $N = L\{A\}$, $\Sigma = \emptyset$ and $\sigma = \{L\{\lambda\}, L\{A\}\} \to \{\lambda\}$. The following table represents $\Sigma^+_{\mathfrak{R}}$.

	$\{\lambda\}$	$\{L\{\lambda\}\}$	$\{L\{A\}\}$	$\{\lambda, L\{\lambda\}\}$	$\{\lambda, L\{A\}\}$	$\{L\{\lambda\}, L\{A\}\}$	$\{\lambda, L\{\lambda\}, L\{A\}\}$
$\{\lambda\}$	×						
$\{L\{\lambda\}\}$	×	×		×			
$\{L\{A\}\}$	×	×	×		×	×	
$\{\lambda, L\{\lambda\}\}$	×	×		×			
$\{\lambda, L\{A\}\}$	×		×		×		
$\{L\{\lambda\}, L\{A\}\}$		×	×			×	
$\{\lambda, L\{\lambda\}, L\{A\}\}$	×	×	×	×	×	×	×

We can see that $\sigma \notin \Sigma^{+}_{\mathfrak{R}}$. However, $\{L\{\lambda\}, L\{A\}\} \to \{\lambda\}$ can be inferred from $\{L\{\lambda\}, L\{A\}\} \to \{L\{\lambda\}\}$, $\{L\{\lambda\}\} \to \{\lambda\} \in \Sigma^{+}_{\mathfrak{R}}$ by the transitivity rule. We conclude that σ can be inferred from Σ using the transitivity rule and $\mathfrak{R}$. $\square$

It is interesting to note that in every of the previous lemmata trivial FDs have been identified as witnesses for the independence of the respective inference rule, i.e., FDs that follow from the empty set of FDs specified.

The previous lemmata prove the following main result. It shows that there is no proper subset of the generalised Armstrong axioms which forms also a complete set of inference rules for the implication of FDs.

Theorem 38. *The generalised Armstrong axioms form a minimal, sound and complete set of inference rules for the implication of FDs in the presence of records, lists, sets and multisets.*

5. Minimal axiomatisations for all combinations

Theorem 38 captured the implication of FDs in the presence of all types considered in this paper. It is now interesting to ask what the minimal axiomatisations for all subsets of the set of all types are. The extended abstract [49] presented an axiomatisation of FDs in the presence of records and sets. The generalised Armstrong Axioms from Definition 17 are in fact already all needed to capture implication in the presence of these two types. The proofs in Section 4 show now that this axiomatisation is also minimal.

Multisets behave similar to sets, in the sense that values on the join of two subattributes are not determined by the individual values on the subattributes. Therefore, the axiomatisation of FDs in the presence of records and multisets is also given by the generalised Armstrong Axioms. Moreover, the proofs in Section 4 are completely analogous, if set-valued attributes are replaced by multiset-valued attributes. Therefore, the axiomatisation is even minimal.

The situation becomes easier if only records and lists are considered. Here, the projections of any tuple on arbitrary subattributes always determine the projection of that tuple on the join of these subattributes. This means that it is sufficient to consider FDs of the form $X \to Y$ where X and Y are subattributes of some nested attribute N. Sets of subattributes are no longer required as all pairs of subattributes are reconcilable. It has been shown in [52] that the implication of FDs can be captured by a generalisation of Armstrong's original axioms. We can therefore summarise the results of our paper in the following theorem.

Theorem 39.

- *The generalised Armstrong Axioms, i.e.,*

$$\frac{}{X \to Y}\; Y \leqslant X, \qquad \frac{X \to Y}{X \to X \sqcup_N Y}, \qquad \frac{X \to Y,\; Y \to Z}{X \to Z}$$

  form a minimal, sound and complete set of inference rules for the implication of FDs in the presence of records, and in the presence of records and lists.
- *Let $\mathcal{T}$ be any non-empty subset of {lists}, sets, multisets} apart from {lists}. The generalised Armstrong Axioms, i.e.,*

$$\frac{}{\mathcal{X} \to \mathcal{Y}}\; \mathcal{Y} \subseteq \mathcal{X}, \qquad \frac{}{\{X\} \to \{Y\}}\; Y \leqslant X, \qquad \frac{\mathcal{X} \to \mathcal{Y}}{\mathcal{X} \to \mathcal{X} \cup \mathcal{Y}},$$

$$\frac{}{\{X, Y\} \to \{X \sqcup_N Y\}}\; X, Y \text{ reconcilable}, \qquad \frac{\mathcal{X} \to \mathcal{Y},\; \mathcal{Y} \to \mathcal{Z}}{\mathcal{X} \to \mathcal{Z}},$$

  form a minimal, sound and complete set of inference rules for the implication of FDs in the presence of records and $\mathcal{T}$.

6. Related work

Dependency theory is a well-studied area of research in the context of the RDM. Excellent surveys are provided in [37,71,74]. The RDM is completely captured by a single application of the record constructor.

The nested RDM [56] has also attracted research on dependency theory, especially on the issue of normalisation [61,63]. The FDs studied in those papers arise from a relational representation of the data assuming a complete unnesting. Take for instance the nested schema $\{Course(Student\text{-}ID, Name)^*\}$ in which for each course the set of participating students is stored, i.e., their student identification number together with their name. A typical FD would be

$$Student\text{-}ID \to Name,$$

i.e., the student identification number uniquely determines the student's name over all courses. FDs in which a set of objects is determined by some object or in which a set of objects determines an object are not considered. An example of such an FD would be

$$Course \to (Student\text{-}ID)^*,$$

where the course determines the set of the identification numbers of its participants. This, however, can be done using record- and set-valued attributes. Consider the nested attribute $Enrolment(Course, Participant\{Student(ID, Name)\})$. The FD above is then specified by

$$Enrolment(Course) \to Enrolment(Participant\{Student(ID)\}).$$

On the other hand, FDs in which inside a set-valued attribute $L\{N\}$ some subattributes of N determine another subattribute of N can be expressed by the previous approaches but are not yet covered by our approach. The previous example suggests for instance to consider the structure of embedded nested attributes such as $Student(ID, Name)$. Then the FD

$$Student(ID) \to Student(Name)$$

does reflect the FD above. The nested RDM is covered by the presence of record- and set-valued attributes.

Next we consider two approaches which have studied FDs in the presence of finite sets. In [42] FDs are defined as *well-defined path expressions* in the presence of records and finite sets. An axiomatisation for the implication of those FDs is provided. However, the FDs do not allow arbitrary nesting, and most importantly, the right-hand side of every FD is always a single path. As the results in this thesis point out the case where the right-hand side is the union of paths is particularly interesting in the presence of sets (the join axiom is only valid in restricted form). FDs of the form

$$\{S\{L(A)\}, S\{L(B)\}\} \to S\{L(A, B)\}$$

cannot be expressed by the approach in [42] as this FD is different from the two trivial FDs

$$\{S\{L(A)\}, S\{L(B)\}\} \to S\{L(A)\} \quad \text{and} \quad \{S\{L(A)\}, S\{L(B)\}\} \to S\{L(B)\}.$$

There are still differences even if we consider only single paths in the right-hand side. Consider for instance the nested attribute $N(L\{K(A, B, C)\}, D)$ together with the FD

$$N(L\{K(A, B)\}) \to N(D),$$

where the set of value pairs on A, B determines the value on D. FDs which are expressible by the approach in [42] are

$$N : [L \to D] \quad \text{and} \quad N : [L : A, L : B \to D]$$

assuming that the labels identify the (embedded) nested attributes. These, however, are both different from

$$N(L\{K(A, B)\}) \to N(D).$$

The first FD corresponds to

$$N(L\{K(A, B, C)\}) \to N(D)$$

and the second corresponds to

$$\{N(L\{K(A)\}), N(L\{K(B)\})\} \to N(D),$$

respectively. On the other hand, in order to express the FD $N : L[A \to B]$ in our context, we need to consider the embedded nested attributes $K(A, B, C)$ where the FD $K(A) \to K(B)$ could be defined. Moreover, attributes in which

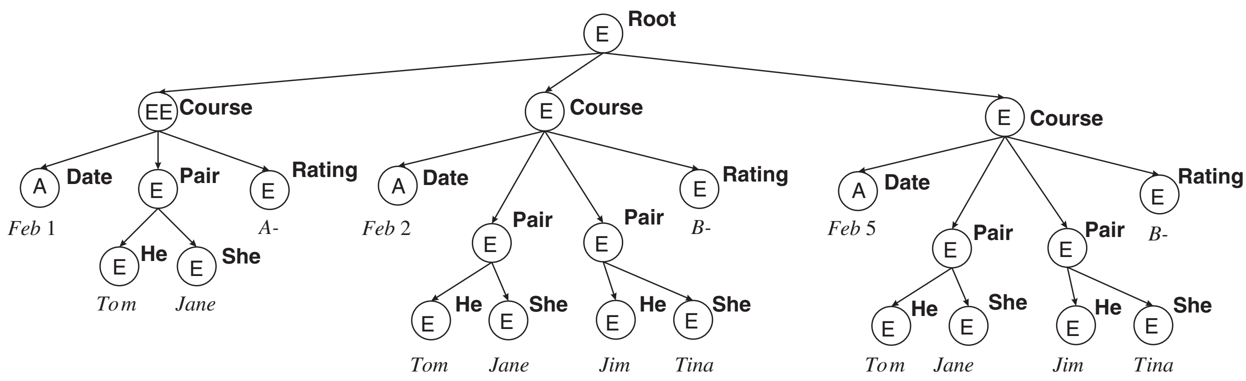

Fig. 9. An XML data tree carrying some functional dependency.

λ occurs are not covered in [42]. In summary, the approach in [42] uses partly the expressiveness of the set constructor, but does not take care of the fact that the extension rule is not valid in the presence of sets. Currently, the expressiveness of our FDs in the presence of null, flat, record- and set-valued attributes is incomparable to the expressiveness of the FDs from [42].

A further approach to defining FDs in the context of the nested RDM is provided in [57]. So-called *null extended FDs* are defined to admit null values and study the relationship between multi-valued dependencies (MVDs) $X \twoheadrightarrow Y$ and FDs $X \rightarrow Y^*$ (here Y refers to the complete unnesting of the relation-valued attribute Y^*), i.e., the interaction of different dependency classes in the context of nesting and unnesting. Null extended FDs are again defined on the basis of paths. FDs from the RDM cannot be expressed. Furthermore, relation-valued attributes can only occur on the right-hand side of null extended FDs. Consider the nested attribute $N = L(A, K\{M(B, S\{C\})\})$ which would be expressed as $A(B(C)^*)^*$ in a slightly simplified nested RDM. Examples for null extended FDs are

$$A \rightarrow (B(C)^*)^* \quad \text{or} \quad AB \rightarrow (C)^*.$$

The last of these is not covered yet by our data model. In order to express the last null extended FD in our context we need to consider combinations of embedded nested attributes, i.e., $L(A, M(B, S\{C\}))$ in this case. Conversely, the FD $L(A, K\{M(B)\}) \rightarrow L(K\{M(S\{C\})\})$ is again not expressible as a null extended FD. The expressiveness of null extended FDs and FDs in the presence of null, flat, record- and set-valued attributes is different.

Most recently, the major research interest is on the model of semi-structured data and XML [1,24]. Work on integrity constraints in the context of XML and object-oriented databases can be found in [5,23,26,38,39,55,70,77–79,83]. The approaches in [5,23,55,70,78,83] are again based on a relational representation of the data, thus resulting again in a different expressiveness from our approach. FDs in [5] are not axiomatisable at all. In order to illustrate the difference to our data model a bit more we look at some examples.

Consider the XML data tree in Fig. 9 containing data on courses organised by the dancing club of the local high school.

The XML document corresponding to this XML data tree is shown in Fig. 10.

It happens that neither gentlemen nor ladies change their dance partners. That is, for every pair in the XML data tree He determines She, and vice versa. Both observations are likely to be called FDs.

Now consider the XML data tree in Fig. 11. It is obvious that the observed FDs do no longer hold. Nevertheless the data stored in this tree is not independent from each other: whenever two courses coincide in all their pairs then they coincide in their rating, too. That is, in every course the set of Pairs determines the Rating. The reason for this might be straightforward. Suppose, during every course each pair is asked whether they enjoyed dancing with each other (and suppose that the answer will not change over time). Afterwards, the average rating is calculated for the course and stored within the XML document. This, in fact, leads to the FD observed in Fig. 11.

Surprisingly, [5,55,78] all introduced the first kind of FDs for XML while the second kind has been neglected so far in the literature on XML. The reason for this is the path-based approach towards functional dependencies used in all three papers. The second kind, however, represents FDs that can be captured using nested attributes. Suppose we have

```
<Root>
  <Course Date="Feb 1">
    <Pair>
      <He>Tom</He>
      <She>Jane</She>
    </Pair>
    <Ranking>A-</Ranking>
  </Course>
  <Course Date="Feb 2">
    <Pair>
      <He>Tom</He>
      <She>Jane</She>
    </Pair>
    <Pair>
      <He>Jim</He>
      <She>Tina</She>
    </Pair>
    <Ranking>B-</Ranking>
  </Course>
  <Course Date="Feb 5">
    <Pair>
      <He>Tom</He>
      <She>Jane</She>
    </Pair>
    <Pair>
      <He>Jim</He>
      <She>Tina</She>
    </Pair>
    <Ranking>B-</Ranking>
  </Course>
</Root>
```

Fig. 10. An XML document corresponding to the XML data tree in Fig. 9.

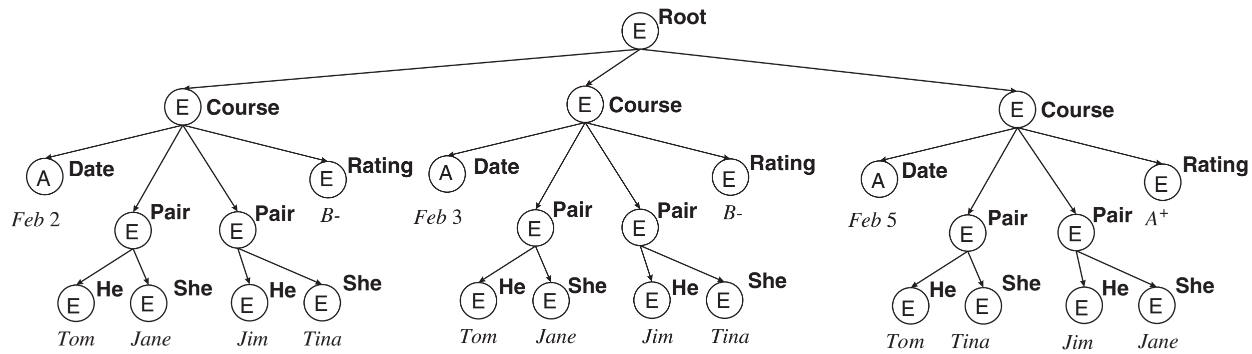

Fig. 11. Another XML data tree still carrying some functional dependency.

the nested attribute

$$Course(Date, Pair\{Partner(He, She)\}, Rating),$$

then the FD above reads as

$$Course(Pair\{Partner(He, She)\}) \rightarrow Course(Rating).$$

In order to capture the first kind of FDs via nested attributes one needs to consider the embedded nested attribute *Partner*(*He*, *She*). In this case the FDs read as $Partner(He) \rightarrow Partner(She)$ and $Partner(She) \rightarrow Partner(He)$. For a graph-oriented approach towards FDs in XML that is based on homomorphisms between subgraphs see [45] and [50].

In order to capture the full expressiveness of XML one will need to consider the union and reference type. Thus, a Kleene-star element definition $\langle !ELEMENT\ X(Y)^*\rangle$ can be represented by the list-valued nested attribute $X[Y]$, a sequence element definition $\langle !ELEMENT\ X(Y_1, \ldots, Y_n)\rangle$ by the record-valued attribute $X(Y_1, \ldots, Y_n)$, and an alternative element definition $\langle !ELEMENT\ X(Y_1|\cdots|Y_n)\rangle$ by $X(Y_1 \oplus \cdots \oplus Y_n)$. Furthermore, as the plus-operator in regular expressions can be expressed by the Kleene-star, an element definition $\langle !ELEMENT\ X(Y)^+\rangle$ can be represented by the record-valued attribute $X(Y, X'[Y])$ with a new label X'. Similarly, optional elements can be expressed by alternatives with empty elements, thus an element definition $\langle !ELEMENT\ X(Y?)\rangle$ will be represented by the union-valued attribute $L(X(Y) \oplus X'(\lambda))$. In order to capture the reference structures in XML documents we may need to consider rational tree attributes. See [32] for fundamental properties of infinite trees. In this case, the subattribute lattice may become infinite.

In summary, our approach based on explicit subattributes deviates significantly from previous approaches in the nested RDM, object-oriented data models and XML, yielding a complementary expressiveness. In particular, the algebraic approach based on a Brouwerian algebra of subattributes is original. The authors are not aware of any other work which deals specifically with list and multiset types in the context of FDs.

7. Conclusion and future work

The work in this paper provides an abstract data model that allows to capture many relevant existing data models according to the types they support. Nested attributes can be generated from flat attributes by various constructions such as records, lists, sets, and multisets. The set of all subattributes of some fixed nested attribute carries the structure of a Brouwerian algebra in which the operations of meet, join and pseudo-difference naturally generalise the set operations of intersection, union and difference from the RDM. Our algebraic approach allows to study various problems generalised from relational dependency theory under one unifying framework which emphasises the impact of the data type rather than the specifics of a particular data model.

In this paper, we have investigated the most common class of dependencies, FDs, in the presence of records, lists, sets and multisets. The main result provides minimal, sound and complete sets of inference rules for the implication of FDs in all combinations of these types which include the record type, i.e., capture at least the RDM. In the presence of records and sets, the expressibility of our FDs is complementary to the expressibility of those that have been studied in previous works on the nested RDM. Our inference rules look very similar to Armstrong's original axioms for FDs in the RDM, even in the presence of multiple types. Besides generalisations of the three original rules, only two new axioms are required to completely capture FDs for all types studied. While the completeness proof for lists is rather straightforward, the cases of set and multiset types require non-trivial combinatorial arguments.

Future work is best explained using Fig. 1. The class of FDs should be studied in the presence of union and reference types which are particularly important for XML [1,24]. The simplicity of the inference rules in Theorem 39 allows us to obtain polynomial-time algorithms for deciding the implication of FDs in the context of various types. This may help to decide the equivalence of sets of dependencies or finding minimal covers for a set of FDs. We intend to extend previous work on normal forms, i.e. syntactically describe well-designed nested attributes with respect to a given set of constraints, and to semantically justify this proposal. This means to formally prove the absence of redundancies and abnormal update behaviour for nested attributes in the normal form proposed. The beginning of this research has already been made in [46] where the Nested List Normal Form (NLNF) has been proposed and justified. NLNF is strictly weaker than a simple extension of Boyce–Codd normal form [19]. Since we used the axiomatisation of FDs in the presence of lists to show the equivalence of NLNF to the absence of redundancies and update anomalies, the axiomatisation in this paper may help to justify normal form proposals for more sophisticated combinations of types. As we have seen in Section 6, our class of FDs deviates from other FDs in the presence of records and sets. The work in [51] proposes therefore a further normal form which is again equivalent to the absence of redundancies and abnormal update behaviour caused by these FDs. The proposed normal form is different from other normal form proposals in the nested RDM [61,63]. The decomposition and synthesis of nested attributes is also subject of future research [13,14,17–19,76].

More classes of relational dependencies are to be studied next, e.g. MVDs, join and inclusion dependencies. The work in [47,52] provide minimal axiomatisations for the classes of MVDs, and FDs and MVDs in the presence of records and lists, thus generalising the work in [15]. Here, the full power of the Brouwerian algebra of subattributes is required since the pseudo-difference operator appears in many of the inference rules. In the presence of lists, the MVD $X \twoheadrightarrow Y$ implies the non-trivial FD $X \to Y \sqcap Y^{\mathcal{C}}$. This is a fundamental and interesting difference to the RDM. A provably-correct polynomial time algorithm for the implication of FDs and MVDs in the presence of records and lists can be found in [48] which naturally generalises the work in [11]. We intend to address normalisation for FDs and MVDs leading to a normal form proposal which is likely to deviate from a simple extension of the well-known fourth normal form [35,36,80,81]. For an excellent overview on classes of relational dependencies see [71].

Finally, a more general treatment in which data dependencies are interpreted as formulae in a suitable logic may result in a successful treatment as in the RDM [37,74].

References

[1] S. Abiteboul, P. Buneman, D. Suciu, Data on the Web: From Relations to Semistructured Data and XML, Morgan Kaufmann Publishers, Los Altos, CA, 2000.
[2] S. Abiteboul, R. Hull, V. Vianu, Foundations of Databases, Addison-Wesley, Reading, MA, 1995.
[3] I. Anderson, Combinatorics of Finite Sets, Oxford Science Publications, The Clarendon Press, Oxford University Press, New York, 1987.
[4] M. Arenas, L. Libkin, An information-theoretic approach to normal forms for relational and XML data, in: Principles of Database Systems (PODS), ACM, New York, NY, 2003, pp. 15–26.
[5] M. Arenas, L. Libkin, A normal form for XML documents, Trans. Database Systems (TODS) 29 (1) (2004) 195–232.
[6] W.W. Armstrong, Dependency structures of database relationships, Inform. Process. (1974) 580–583.
[7] W.W. Armstrong, Y. Nakamura, P. Rudnicki, Armstrong's axioms, J. Formalized Math. 14.
[8] M. Atkinson, F. Bancilhon, D. DeWitt, K. Dittrich, D. Maier, S. Zdonik, The object-oriented database system manifesto, in: Proc. Internat. Conf. on Deductive and Object-Oriented Databases, 1989, pp. 40–57.
[9] J. Banatre, D. Le Metayer, Programming by multiset transformation, Comm. ACM 36 (1) (1993) 98–111.
[10] C. Batini, S. Ceri, S.B. Navathe, Conceptual Database Design: An Entity-Relationship Approach, Benjamin Cummings, Menlo Park, CA, 1992.
[11] C. Beeri, On the membership problem for functional and multivalued dependencies in relational databases, Trans. Database Systems (TODS) 5 (3) (1980) 241–259.
[12] C. Beeri, A formal approach to object-oriented databases, Data Knowledge Eng. 5 (4) (1990) 353–382.
[13] C. Beeri, P.A. Bernstein, Computational problems related to the design of normal form relational schemata, Trans. Database Systems (TODS) (1979) 30–59.
[14] C. Beeri, P.A. Bernstein, N. Goodman, A sophisticate's introduction to database normalization theory, in: Proc. Internat. Conf. on Very Large Data Bases (VLDB), 1978, pp. 113–124.
[15] C. Beeri, R. Fagin, J.H. Howard, A complete axiomatization for functional and multivalued dependencies in database relations, in: Internat. Conf. on Management of Data (SIGMOD), ACM, New York, NY, 1977, pp. 47–61.
[16] C. Beeri, A.O. Mendelzon, Y. Sagiv, J.D. Ullman, Equivalence of relational database schemes, SIAM J. Comput. 10 (2) (1981) 352–370.
[17] P.A. Bernstein, Normalisation and functional dependencies in the relational data base model, Technical Report CSRG-60, University of Toronto, 1975.
[18] P.A. Bernstein, Synthesizing third normal form relations from functional dependencies, Trans. Database Systems (TODS) 1 (1976) 277–298.
[19] P.A. Bernstein, N. Goodman, What does Boyce–Codd normal form do, in: Proc. Internat. Conf. on Very Large Data Bases (VLDB), 1980, pp. 245–259.
[20] G. Berry, G. Boudol, The chemical abstract machine, Theoret. Comput. Sci. 96 (1992) 217–248.
[21] J. Biskup, Database schema design theory: achievements and challenges, in: Information Systems and Data Management, Lecture Notes in Computer Science, vol. 1066, Springer, Berlin, 1995, pp. 14–44.
[22] J. Biskup, Achievements of relational database schema design theory revisited, in: Semantics in Databases, Lecture Notes in Computer Science, vol. 1358, Springer, Berlin, 1998, pp. 29–54.
[23] M.F. Bommel, G.E. Weddell, Reasoning about equations and functional dependencies on complex objects, Trans. Knowledge Data Eng. 6 (3) (1994) 455–469.
[24] T. Bray, J. Paoli, C.M. Sperberg-McQueen, E. Maler, F. Yergeau, Extensible markup language (XML) 1.0, third ed., W3C Recommendation 04 February 2004. ⟨http://www.w3.org/TR/2004/REC-xml-20040204/⟩ (2004).
[25] F. Bry, P. Kröger, A computational biology database digest: data, data analysis, and data management, Distributed Parallel Databases 13 (1) (2003) 7–42.
[26] P. Buneman, W. Fan, J. Siméon, S. Weinstein, Constraints for semi-structured data and XML, SIGMOD Record 30 (1) (2001) 47–54.
[27] C. Calude, G. Paun, G. Rozenberg, A. Salomaa (Eds.), Multiset processing, mathematical, computer science, and molecular computing points of view, in: Workshop on Multiset Processing, WMP 2000, Curtea de Arges, Romania, August 21–25, 2000, Lecture Notes in Computer Science, vol. 2235, Springer, Berlin, 2001.
[28] P.P. Chen, The entity-relationship model: towards a unified view of data, Trans. Database Systems (TODS) 1 (1976) 9–36.
[29] P.P. Chen, English sentence structure and entity-relationship diagrams, Inform. Sci. 29 (1983) 127–149.
[30] E.F. Codd, Further normalization of the database relational model, in: Courant Computer Science Symposia, vol. 6: Data Base Systems, Prentice-Hall, Englewood Cliffs, NJ, 1972, pp. 33–64.
[31] E.F. Codd, Recent investigations in relational database system, in: Proc. IFIP Conf., 1974, pp. 1017–1021.
[32] B. Courcelle, Fundamental properties of infinite trees, Theoret. Comput. Sci. 25 (1983) 95–169.
[33] A. Dovier, A. Policriti, G. Rossi, A uniform axiomatic view of lists, multisets, and sets, and the unification algorithm, Fund. Inform. 36 (2/3) (1998) 201–234.
[34] K. Engel, Sperner Theory, Cambridge University Press, Cambridge, 1997.
[35] R. Fagin, Multivalued dependencies and a new normal form for relational databases, Trans. Database Systems (TODS) 2 (3) (1977) 262–278.
[36] R. Fagin, A normal form for relational databases that is based on domains and keys, Trans. Database Systems (TODS) 6 (3) (1981) 387–415.
[37] R. Fagin, M.Y. Vardi, The theory of data dependencies: a survey, in: Mathematics of Information Processing: Proc. Symp. in Applied Mathematics, American Mathematical Society, Providence, RI, 1986, pp. 19–71.
[38] W. Fan, L. Libkin, On XML integrity constraints in the presence of DTDs, J. ACM 49 (3) (2002) 368–406.
[39] W. Fan, J. Siméon, Integrity constraints for XML, J. Comput. System Sci. 66 (1) (2003) 254–291.
[40] D.H. Fishman, D. Beech, H.P. Cate, E.F. Chow, T. Connors, J.W. Davis, N. Derret, C.G. Hoch, W. Kent, P. Lyngbæk, B. Mahbod, M.-A. Neimat, T.A. Ryan, M.-C. Shan, IRIS: an object-oriented database management system, Trans. Office Inform. Systems 5 (1) (1987) 48–69.

[41] G. Gardarin, J.-P. Cheiney, G. Kiernan, D. Pastre, H. Stora, Managing complex objects in an extensible relational DBMS, in: Proc. Internat. Conf. on Very Large Data Bases, Morgan Kaufmann, Los Altos, CA, 1989, pp. 55–65.
[42] C.S. Hara, S.B. Davidson, Reasoning about nested functional dependencies, in: Principles of Database Systems (PODS), ACM, New York, 1999, pp. 91–100.
[43] S. Hartmann, On the implication problem for cardinality constraints and functional dependencies, Ann. Math. Artificial Intelligence 33 (2001) 253–307.
[44] S. Hartmann, Decomposing relationship types by pivoting and schema equivalence, Data Knowledge Eng. 39 (2001) 75–99.
[45] S. Hartmann, S. Link, More functional dependencies for XML, in: Advances in Databases and Information Systems, Seventh East European Conf. ADBIS 2003, Dresden, Germany, September 3–6, Lecture Notes in Computer Science, vol. 2798, Springer, Berlin, 2003, pp. 355–369.
[46] S. Hartmann, S. Link, Normalisation in the presence of lists, in: 15th Australasian Database Conf. (ADC), Conf. in Research and Practice in Information Technology, vol. 27, Australian Computer Society, 2004, pp. 53–64.
[47] S. Hartmann, S. Link, Multi-valued dependencies in the presence of lists, in: Proc. 23rd Internat. Conf. on Principles of Database Systems (PODS), ACM, New York, NY, 2004, pp. 330–341.
[48] S. Hartmann, S. Link, A membership algorithm for functional and multi-valued dependencies in the presence of lists, Electron. Notes Theoret. Comput. Sci. (ENTCS) 91 (2004) 171–194.
[49] S. Hartmann, S. Link, On functional dependencies in advanced data models, Electron. Notes Theoret. Comput. Sci. (ENTCS) 84.
[50] S. Hartmann, S. Link, M. Kirchberg, A subgraph-based approach towards functional dependencies for XML, in: Proc. Seventh World-Multiconf. on Systemics, Cybernetics and Informatics (SCI), Computer Science and Engineering II, vol. IX, Orlando, Florida, USA, July 27–30, 2003, pp. 200–205.
[51] S. Hartmann, S. Link, K.-D. Schewe, A new normal form for conceptual databases, in: Y. Kiyoki, E. Kawaguchi, H. Jaakkola, H. Kangassalo (Eds.), Information Modelling and Knowledge Bases XV, 13th European–Japanese Conf. on Information Modelling and Knowledge Bases (EJC 2003), Kitakyushu, Japan, June 3–6, 2003, Frontiers in Artificial Intelligence and Applications, vol. 105, IOS Press, 2004, pp. 93–110.
[52] S. Hartmann, S. Link, K.-D. Schewe, Reasoning about functional and multi-valued dependencies in the presence of lists, in: Third Internat. Symp. on Foundations of Information and Knowledge Systems (FoIKS), Lecture Notes in Computer Science, vol. 2942, Springer, Berlin, 2004, pp. 134–154.
[53] R. Hull, R. King, Semantic database modeling: survey, applications and research issues, ACM Comput. Surveys 19(3).
[54] G. Lamperti, M. Melchiori, M. Zanella, On multisets in database systems, in: Workshop on Multiset Processing (WMP), Lecture Notes in Computer Science, vol. 2235, Springer, Berlin, 2000, pp. 147–216.
[55] M. Lee, T. Ling, W. Low, Designing functional dependencies for XML, in: Advances in Database Technology—EDBT 2002, Eighth Internat. Conf. on Extending Database Technology, Prague, Czech Republic, March 25–27, Lecture Notes in Computer Science, vol. 2287, Springer, Berlin, 2002, pp. 124–141.
[56] M. Levene, The Nested Universal Relation Database Model, Springer, Berlin, 1992.
[57] M. Levene, G. Loizou, Semantics for null extended nested relations, Trans. Database Systems (TODS) 18 (3) (1993) 414–459.
[58] J. Li, S. Ng, L. Wong, Bioinformatics adventures in database research, in: Proc. Internat. Conf. on Database Theory (ICDT), Lecture Notes in Computer Science, vol. 2572, Springer, Berlin, 2002, pp. 31–46.
[59] D. Maier, Minimum covers in relational database model, J. ACM 27 (4) (1980) 664–674.
[60] J.C.C. McKinsey, A. Tarski, On closed elements in closure algebras, Ann. Math. 47 (1946) 122–146.
[61] W.Y. Mok, Y.K. Ng, D.W. Embley, A normal form for precisely charachterizing redundancy in nested relations, Trans. Database Systems TODS 21 (1996) 77–106.
[62] S. Naqvi, S. Tsur, A Logical Language for Data and Knowledge Bases, Computer Science Press, Rockville, MD, 1989.
[63] Z.M. Özsoyoglu, L.Y. Yuan, A new normal form for nested relations, Trans. Database Systems TODS 12 (1987) 111–136.
[64] J. Paredaens, P. De Bra, M. Gyssens, D. Van Gucht, The Structure of the Relational Database Model, Springer, Berlin, 1989.
[65] J. Richardson, Supporting lists in a datamodel, in: Proc. Internat. Conf. on Very Large Data Bases (VLDB), 1992, pp. 127–192.
[66] K.-D. Schewe, B. Thalheim, Fundamental concepts of object oriented databases, Acta Cybernet. 11 (4) (1993) 49–85.
[67] M.H. Scholl, H.-J. Schek, A relational object model, in: Proc. of Internat. Conf. on Database Theory (ICDT), Lecture Notes Series, Springer, Berlin, 1990, pp. 89–105.
[68] P. Seshadri, M. Livny, R. Ramakrishnan, The design and implementation of sequence database system, in: Proc. Internat. Conf. on Very Large Data Bases (VLDB), 1996, pp. 99–110.
[69] D. Suciu, On database theory and XML, SIGMOD Record 30 (3) (2001) 39–45.
[70] Z. Tari, J. Stokes, S. Spaccapietra, Object normal forms and dependency constraints for object-oriented schemata, Trans. Database Systems (TODS) 22 (1997) 513–569.
[71] B. Thalheim, Dependencies in Relational Databases, Teubner, Stuttgart, 1991.
[72] B. Thalheim, Entity-Relationship Modeling: Foundations of Database Technology, Springer, New York, NY, 2000.
[73] A.M. Tjoa, L. Berger, Transformation of requirement specifications expressed in natural language into an EER model, in: Entity-Relationship Approach, Lecture Notes in Computer Science, vol. 823, Springer, Berlin, 1993, pp. 206–217.
[74] M.Y. Vardi, Fundamentals of dependency theory, in: E. Börger (Ed.), Trends in Theoretical Computer Science, Computer Science Press, Rockville, MD, 1987, pp. 171–224.
[75] V. Vianu, A web odyssey: from Codd to XML, in: PODS, ACM, New York, NY, 2001, pp. 1–15.
[76] M. Vincent, The semantic justification for normal forms in relational database design, Ph.D. Thesis, Monash University, Melbourne, Australia, 1994.
[77] M. Vincent, J. Liu, Multivalued dependencies in XML, in: British National Conf. on Databases, Lecture Notes in Computer Science, vol. 2712, Springer, Berlin, 2003, pp. 4–18.

[78] M. Vincent, J. Liu, Functional dependencies for XML, in: M.O.X. Zhou, Y. Zhang (Eds.), Web Technologies and Applications: Fifth Asia-Pacific Web Conf. APWeb 2003, Xian, China, April 23–25, 2003. Proc., Lecture Notes in Computer Science, vol. 2642, Springer, Berlin, 2003, pp. 22–34.
[79] M. Vincent, J. Liu, C. Liu, A redundancy free 4NF for XML, in: Proc. XML Database Symp., Lecture Notes in Computer Science, vol. 2824, Springer, Berlin, 2003, pp. 254–266.
[80] M. Vincent, B. Srinivasan, Redundancy and the justification of fourth normal form in relational databases, Internat. J. Found. Comput. Sci. 4 (4) (1993) 355–365.
[81] M. Vincent, B. Srinivasan, Update anomalies and the justification of fourth normal form in relational databases, Inform. Sci. 81 (1994) 87–102.
[82] G. Vossen, A new characterization of FD implication with an application to update anomalies, Inform. Process. Lett. 29 (1988) 131–135.
[83] G.E. Weddell, Reasoning about functional dependencies generalized for semantic data models, Trans. Database Systems (TODS) 17 (1) (1992) 32–64.