# Functional Dependencies over XML Documents with DTDs

Sven Hartmann[*], Sebastian Link[*], and Klaus-Dieter Schewe[*]

**Abstract**

In this article an axiomatisation for functional dependencies over XML documents is presented. The approach is based on a representation of XML document type definitions (or XML schemata) by nested attributes using constructors for records, disjoint unions and lists, and a particular null value, which covers optionality. Infinite structures that may result from referencing attributes in XML are captured by rational trees. Using a partial order on nested attributes we obtain non-distributive Brouwer algebras. The operations of the Brouwer algebra are exploited in the soundness and completeness proofs for derivation rules for functional dependencies.

**Keywords:** eXtensible Markup Language, nested attributes, subattributes, rational trees, functional dependencies, axiomatisation

## 1 Introduction

Over the last decade the eXtensible Markup Language (XML) [5] has attracted a lot of attention in research and practice. Its spectrum of usage spreads from data exchange on the web to a direct use as a data model. In fact, the language shows a lot of similarities to semi-structured data [1] and to object-oriented databases [12].

The treatment of XML as a data model requires re-investigating core problems of database theory. Therefore, it is no surprise that database dependency theory [13] has recently started a revival in the context of XML. The research interest first focused on the classes of keys [4, 11] and functional dependencies [3, 16, 18], which represent the most common and at the same time easiest class of dependencies.

However, the problem is still not completely solved. The major drawback of the work by Arenas, Fan and Libkin and similarly Vincent and Liu is the restriction to a relational representation of XML documents. That is, XML documents are considered as some sets of (generalised) tuples, which then can be treated analogously to the relational model. However, it is possible to formulate functional dependencies on XML documents that are not preserved by the relational representation. In

[*]Massey Information Science Research Centre, Private Bag 11222, Palmerston North, New Zealand. E-mail: `[s.hartmann|s.link|k.d.schewe]@massey.ac.nz`

other words, these theories are adequate as long as we only deal with functional dependencies that can be expressed on a relational representation of XML documents. Going beyond this restricted class of functional dependencies requires an extended theory or a different approach to the problem.

Our own work in this area originates from a more classical approach dealing with dependencies in higher-order data models such as the higher-order Entity-Relationship model (HERM) [14] or the object-oriented data model (OODM) [12]. The basic idea is to consider nested attributes that can be built from constructors for records, sets, lists, etc. Furthermore, our first interest is devoted to the logical and mathematical foundations of dependency theory, i.e. we first address problems of axiomatisation, number of possible dependencies, complexity of closure building, etc.

Using just the record- and set-constructors we obtained an axiomatisation in [6], extended in [7]. Additional constructors for lists, multisets and disjoint unions have been handled in [10]. Unfortunately, the presence of the union-constructor, in particular in connection with the set-constructor, requires an extension of the theory to weak functional dependencies, i.e. disjunctions of functional dependencies. Rather astonishingly, among the three "bulk" constructors the list-constructor is the easiest one. So far it is the only part of the theory that could be generalised to multi-valued dependencies [9]. Other work on multi-valued dependencies for XML [17] is again "relationally minded".

In this article we extend our theory of functional dependencies to XML documents. We show how to represent XML elements by nested attributes. In particular, we represent the Kleene-star by the list-constructor, i.e. we have order and duplicates. In our theory it is also possible to use the multiset- or set-constructor instead, thus neglecting order or duplicates. We may also treat all three "bulk" constructors together. However, as this would blow up the article we made the choice to restrict ourselves to only the easiest of the bulk constructors. A glimpse of the necessary extensions for the other two bulk constructors can be obtained from [10].

In any case the combination of a bulk constructor with the union-constructor is only satisfactory, if some form of restructuring is taken into account. The early work in [2] handles only the set-constructor, but even for this the theory would be equivalent to restricting the union-constructor in a way that it can only occur as the outermost constructor. This is insufficient, if subattributes are considered. Therefore, we use an extended form of restructuring.

However, in order to fully capture functional dependencies over XML documents we have to face two major extensions:

1. We have to consider rational tree attributes, which result from reference structures in XML documents. We will see that the extension arising from this problem is not severe. The major observation is that the subattribute lattice becomes infinite, but this does not affect the derivation of dependencies. Note that all previous work on functional dependencies for XML including [3] neglect references.

2. We have to consider functional dependencies on embedded elements. These dependencies can be "lifted" through the constructors, i.e. they induce functional dependencies on complete XML documents. Such dependencies have not been considered in our previous work. However, the "lifting rules" became already indispensable in the presence of the union-constructor, as this constructor leads to axioms on embedded structures [10].

In other words, this article takes a reasonable fragment of the theory from [10] and extends it with respect to these two problems. The result is a quite uniform axiomatisation for functional documents over XML documents.

In the remainder of the article we first investigate the relationship between XML documents and nested attributes in Section 2. We show how to map the regular expressions in XML document type definitions to the attribute constructors. Furthermore, we extend nested attributes by rational trees and use them to represent the infinite structures that may arise from references in XML documents. We then define a partial order on nested attributes and show that the set of subattributes of a given attribute forms nearly a Brouwer algebra — however, distributivity does not hold.

In Section 3 we introduce functional dependencies and first prove the soundness of some derivation rules for them. These soundness rules imply properties of closures, i.e. sets of subattributes that depend functionally on a given set of subattributes. This leads to the notion of "strong higher-level ideal" or SHL-ideal for short. We show a central theorem about such SHL-ideals, which states that we can always find two values in the associated domain that coincide exactly on a given SHL-ideal. This theorem is indeed central for the proof of the completeness of the derivation rules. The completeness theorem will be the major result of this article.

## 2 XML and Nested Attributes

In this section we define our extended model of nested attributes including rational tree attributes. We show how to use these attributes to represent XML document type definitions. Finally, we look a bit closer into the structure of sets of subattributes and show that we obtain non-distributive Brouwer algebras.

### 2.1 Elements in XML and Constructors

The structure of XML documents is prescribed by a document type definition (DTD) [1] or (almost equivalently) by an XML schema. Basically, such a DTD is a collection of element definitions, where each element is defined by a regular expression made out of element names and a single base domain *PCDATA*. Without loss of generality we may assume to have more than one domain. Then we can isolate those element definitions that lead only to domains. These elements can be represented by simple attributes.

**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 *simple attributes.*

For all other element definitions we may assume without loss of generality — just spend a few more element names, if necessary — that they are normalised in the sense that they only contain element names and no domains, and they only use exactly one of the constructors for sequences, Kleene-star or alternative.

Then they can be represented as nested attributes as defined next. We use a set $\mathcal{L}$ of labels, and tacitly assume that the symbol $\lambda$ is neither a simple attribute nor a label, i.e. $\lambda \notin \mathcal{U} \cup \mathcal{L}$, and that simple attributes and labels are pairwise different, 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 $\mathcal{N}$ of *nested attributes* (over $\mathcal{U}$ and $\mathcal{L}$) is the smallest set with $\lambda \in \mathcal{N}$, $\mathcal{U} \subseteq \mathcal{N}$, and satisfying the following properties:

- for $X \in \mathcal{L}$ and $X'_1, \dots, X'_n \in \mathcal{N}$ we have $X(X'_1, \dots, X'_n) \in \mathcal{N}$;
- for $X \in \mathcal{L}$ and $X' \in \mathcal{N}$ we have $X[X'] \in \mathcal{N}$;
- for $X_1, \dots, X_n \in \mathcal{L}$ and $X'_1, \dots, X'_n \in \mathcal{N}$ we have $X_1(X'_1) \oplus \dots \oplus X_n(X'_n) \in \mathcal{N}$.

We call $\lambda$ a *null attribute*, $X(X'_1, \dots, X'_n)$ a *record attribute*, $X[X']$ a *list attribute*, and $X_1(X'_1) \oplus \dots \oplus X_n(X'_n)$ a *union attribute*. As record and list attributes have a unique leading label, say $X$, we often write simply $X$ to denote the attribute.

Thus, a Kleene-star element definition $\langle$!ELEMENT $X(Y)^*\rangle$ will be represented by the nested attribute $X[Y]$, a sequence element definition $\langle$!ELEMENT $X(Y_1, \dots, Y_n)\rangle$ by $X(Y_1, \dots, Y_n)$, and an alternative element definition $\langle$!ELEMENT $X(Y_1 \mid \cdots \mid Y_n)\rangle$ by $X(X_1(Y_1) \oplus \dots \oplus X_n(Y_n))$ with some new invented labels $X_1, \dots, X_n$. Furthermore, as the plus-operator in regular expressions can be expressed by the Kleene-star, an element definition $\langle$!ELEMENT $X(Y)^+\rangle$ will be represented by the nested attribute $X(Y, X'[Y])$ with some new invented label $X'$. Similarly, optional elements can be expressed as alternatives with empty elements, thus an element definition $\langle$!ELEMENT $X(Y?)\rangle$ will be represented by the nested attribute $X(Y) \oplus X'(\lambda)$.

We can now extend the association *dom* from simple to nested attributes, i.e. for each $X \in \mathcal{N}$ we will define a set of values $dom(X)$.

**Definition 3.** For each nested attribute $X \in \mathcal{N}$ we get a *domain* $dom(X)$ as follows:

- $dom(\lambda) = \{\top\}$;
- $dom(X(X'_1, \dots, X'_n)) = \{(X_1 : v_1, \dots, X_n : v_n) \mid v_i \in dom(X'_i) \text{ for } i = 1, \dots, n\}$ with labels $X_i$ for the attributes $X'_i$;
- $dom(X[X']) = \{[v_1, \dots, v_n] \mid v_i \in dom(X') \text{ for } i = 1, \dots, n\}$, i.e. each element in $dom(X[X'])$ is a finite list with elements in $dom(X')$;

- $dom(X_1(X_1') \oplus \cdots \oplus X_n(X_n')) = \{(X_i : v_i) \mid v_i \in dom(X_i') \text{ for } i = 1, \dots, n\}$.

Hence, each element in a DTD can be represented by a nested attribute. An XML document is then represented by a value $v \in dom(X)$ of the nested attribute $X$ that represents the root. In the following we assume without loss of generality — rename, if necessary — that labels are used only once in a representing nested attribute. In this way we may identify labels with nested attributes labelled by them.

## 2.2 Attributes in XML and Rational Trees

Besides element definitions a DTD also contains attribute definitions. Attributes are associated with elements. Neglecting some of the syntactic sugar, we basically have three types of attributes:

- attributes with domain *CDATA*, which can be represented again by simple attributes;
- attributes with domain *ID*, which can be ignored;
- attributes with domain *IDREF* or *IDREFS*, which can be replaced by the label, or the list of labels, respectively, of the referenced elements.

More formally, we extend our Definition 2 of nested attributes by adding $\mathcal{L} \subseteq \mathcal{N}$. We say that a label $Y \in \mathcal{L}$ occurring inside a nested attribute $X$, is a *defining label* iff it is introduced by one of the three cases in Definition 2. Otherwise it is a *referencing label*. We require that each label $Y$ appears at most once as a defining label in a nested attribute $X$, and that each referencing label also occurs as a defining label. In other words, if we represent a nested attribute by a labelled tree, a defining label is the label of a non-leaf node, and a referencing label is the label of a leaf node.

Using labels we can subsume the attributes of an element in the element definition using a sequence constructor. Attributes with domain *CDATA* will be represented by simple attributes, attributes with domain *IDREF* will be represented by the label of the referenced element, and attributes with domain *IDREFS* will be represented by the list of labels of the referenced elements.

We still have to extend Definition 3. For this assume $X \in \mathcal{N}$ and let $Y$ be a referencing label in $X$. If we replace $Y$ by the nested attribute that is defined by $Y$ within $X$, we call the result an *expansion* of $X$. Note that in such an expansion a label may now appear more than once as a defining label, but all the nested attributes defined by a label can be identified, as the corresponding sets of expansions are identical.

In order to define domains assume set of *label variables* $\psi(Y)$ for each $Y \in \mathcal{L}$. Then for each expansion $X'$ of a nested attribute $X$ we define $dom(X')$ as in Definition 3 with the following modifications:

- for a referencing label $Y$ we take $dom(Y) = \psi(Y)$;

- for a label $Y$ defining the nested attribute $Y'$ take $dom(Y) = \{y : v \mid y \in \psi(Y), v \in dom(Y')\}$;
- allow only such values $v$ in $dom(X')$, for which the values of referencing labels also occur inside $v$ exactly once at the position of a defining label.

Finally, define $dom(X) = \bigcup_{X'} dom(X')$, where the union spans over all expansions $X'$ of $X$.

## 2.3 Subattributes

In classical dependency theory for the relational model we considered the powerset $\mathcal{P}(R)$ for a relation schema $R$, which is a Boolean algebra with order $\subseteq$. We have to generalise this for nested attributes starting with a partial order $\geq$. However, this partial order will be defined on equivalence classes of attributes. We will identify nested attributes, if we can identify their domains.

**Definition 4.** $\equiv$ is the smallest *equivalence relation* on $\mathcal{N}$ satisfying the following properties:

- $\lambda \equiv X()$;
- $X(X'_1, \dots, X'_n) \equiv X(X'_1, \dots, X'_n, \lambda)$;
- $X(X'_1, \dots, X'_n) \equiv X(X'_{\sigma(1)}, \dots, X'_{\sigma(n)})$ for any permutation $\sigma$;
- $X_1(X'_1) \oplus \dots \oplus X_n(X'_n) \equiv X_{\sigma(1)}(X'_{\sigma(1)}) \oplus \dots \oplus X_{\sigma(n)}(X'_{\sigma(n)})$ for any permutation $\sigma$;
- $X(X'_1, \dots, X'_n) \equiv X(Y_1, \dots, Y_n)$ iff $X'_i \equiv Y_i$ for all $i = 1, \dots, n$;
- $X_1(X'_1) \oplus \dots \oplus X_n(X'_n) \equiv X_1(Y_1) \oplus \dots \oplus X_n(Y_n)$ iff $X'_i \equiv Y_i$ for all $i = 1, \dots, n$;
- $X[X'] \equiv X[Y]$ iff $X' \equiv Y$;
- $X(X'_1, \dots, Y_1(Y'_1) \oplus \dots \oplus Y_m(Y'_m), \dots, X'_n) \equiv$
  $Y_1(X'_1, \dots, Y'_1, \dots, X'_n) \oplus \dots \oplus Y_m(X'_1, \dots, Y'_m, \dots, X'_n)$;
- $X[X_i(X'_i)] \equiv X(X_i[X'_i])$.

Basically, the equivalence definition (apart from the last case) states that $\lambda$ in record attributes can be added or removed, and that order in record and union attributes does not matter. The last case in Definition 4 covers an obvious restructuring rule, which was already introduced in [2].

In the following we identify $\mathcal{N}$ with the set $\mathcal{N}/_{\equiv}$ of equivalence classes. In particular, we will write $=$ instead of $\equiv$, and in the following definition we should say that $Y$ is a subattribute of $X$ iff $\tilde{X} \geq \tilde{Y}$ holds for some $\tilde{X} \equiv X$ and $\tilde{Y} \equiv Y$.

**Definition 5.** For $X, Y \in \mathcal{N}$ we say that $Y$ is a *subattribute* of $X$, iff $X \geq Y$ holds, where $\geq$ is the smallest partial order on $\mathcal{N}$ satisfying the following properties:

- $X \geq \lambda$ for all $X \in \mathcal{N}$;
- $X \geq X'$ for all expansions $X'$ of $X$;
- $X(Y_1, \dots, Y_n) \geq X(X'_{\sigma(1)}, \dots, X'_{\sigma(m)})$ for some injective $\sigma : \{1, \dots, m\} \to \{1, \dots, n\}$ and $Y_{\sigma(i)} \geq X'_{\sigma(i)}$ for all $i = 1, \dots, m$;
- $X_1(Y_1) \oplus \cdots \oplus X_n(Y_n) \geq X_{\sigma(1)}(X'_{\sigma(1)}) \oplus \cdots \oplus X_{\sigma(n)}(X'_{\sigma(n)})$ for some permutation $\sigma$ and $Y_i \geq X'_i$ for all $i = 1, \dots, n$;
- $X[Y] \geq X[X']$ iff $Y \geq X'$;
- $X[X_1(X'_1) \oplus \cdots \oplus X_n(X'_n)] \geq X(X_1[X'_1], \dots, X_n[X'_n])$;
- $X[X_1(X'_1) \oplus \cdots \oplus X_k(X'_k)] \geq X[X_1(X'_1) \oplus \cdots \oplus X_\ell(X'_\ell)]$ for $k \geq \ell$;
- $X(X_{i_1}[\lambda], \dots, X_{i_k}[\lambda]) \geq X_{\{i_1, \dots, i_k\}}[\lambda]$.

Obviously, $X \geq Y$ induces a projection map $\pi^X_Y : \mathit{dom}(X) \to \mathit{dom}(Y)$. For $X \equiv Y$ we have $X \geq Y$ and $Y \geq X$ and the projection maps $\pi^X_Y$ and $\pi^Y_X$ are inverse to each other.

Note that the last three cases in Definition 5 covers the restructuring for lists of unions, which needs some more explanation. Obviously, if we are given a list of elements labelled with $X_1, \dots, X_n$, we can take the individual sublists – preserving the order – that contain only those elements labelled by $X_i$ and build the tuple of these lists. In this case we can turn the label into a label for the whole sublist. This explains the third to last subattribute relationship. In case $n = 1$ this is subsumed by the last equivalence in Definition 4.

Using the subattribute relationship for record attributes we obtain

$$X(X_1[Y_1], \dots, X_n[Y_n]) \geq X(X_{i_1}[Y_{i_1}], \dots, X_{i_k}[Y_{i_k}])$$

for $\{i_1, \dots, i_k\} \subseteq \{1, \dots, n\}$. But then also

$$X[X_{i_1}(Y_{i_1}), \dots, X_{i_k}(Y_{i_k})] \geq X(X_{i_1}[Y_{i_1}], \dots, X_{i_k}[Y_{i_k}])$$

holds as already explained. It is therefore natural to require the second to last property. It just means that a list with elements labelled by $X_1, \dots, X_k$ can be mapped to the sublist – preserving the order – that contains only the elements with labels $X_1, \dots, X_\ell$. We may of course take any subset of the labels here, but this is already captured by the possibility to permute the components in a union attribute.

In a list we can also map each element to $\top$, the unique element in $\mathit{dom}(\lambda)$. In fact, the subattribute of the form $X[\lambda]$ only counts the number of elements in the list. This is not affected by first separating the list according to labels, so we obtain the last subattribute relationship.

However, restructuring requires some care with labels. If we simply reused the label $X$ in the last property in Definition 5, we would obtain

$$X[X_1(X_1') \oplus X_2(X_2')] \geq X(X_1[X_1'], X_2[X_2']) \geq X(X_1[X_1']) \geq X(X_1[\lambda]) \geq X[\lambda].$$

However, the last step here is wrong, as the left hand side refers to the length of the sublist containing the elements with label $X_1$, whereas the right hand side refers to the length of the whole list, i.e. elements have labels $X_1$ or $X_2$. No such mapping can be claimed. In fact, what we really have to do is to mark the list label in an attribute of the form $X[X_1(X_1') \oplus \cdots \oplus X_n(X_n')]$ to indicate the inner union attribute, i.e. we should use $X_{\{1,\dots,n\}}$ (or even $X_{\{X_1,\dots,X_n\}}$) instead of $X$. Then the second to last restructuring property in Definition 5 would become

$$X_{\{1,\dots,k\}}[X_1(X_1') \oplus \cdots \oplus X_k(X_k')] \geq X_{\{1,\dots,\ell\}}[X_1(X_1') \oplus \cdots \oplus X_\ell(X_\ell')].$$

However, as long as we are not dealing with subattributes of the form $X_{\{1,\dots,k\}}[\lambda]$, the additional index does not add any information and thus can be omitted to increase readability. In the last restructuring property in Definition 5, however, the index is needed.

Further note that due to the restructuring rules in Definitions 4 and 5 we may have the case that a record attribute is a subattribute of a list attribute. This allows us to assume that the union-constructor only appears inside a list-constructor or as the outermost constructor. This will be frequently exploited in our proofs.

We use the notation $\mathcal{S}(X) = \{Z \in \mathcal{N} \mid X \geq Z\}$ to denote the *set of subattributes* of a nested attribute $X$. In the next subsection we will take a closer look into the structure of $\mathcal{S}(X)$.

Figure 1 shows the subattributes of $X[X_1(A) \oplus X_2(B)]$ together with the relation $\geq$ on them. Note that the subattribute $X[\lambda]$ would not occur, if we only considered the record-structure, whereas other subattributes such as $X(X_1[\lambda])$ would not occur, if we only considered the list-structure. This is a direct consequence of the restructuring rules.

Let us now investigate the structure of $\mathcal{S}(X)$. We will show that we obtain a non-distributive Brouwer algebra, i.e. a non-distributive lattice with relative pseudo-complements. A lattice $\mathcal{L}$ with zero and one, partial order $\leq$, join $\sqcup$ and meet $\sqcap$ is said to have *relative pseudo-complements* iff for all $Y, Z \in \mathcal{L}$ the infimum $Y \leftarrow Z = \bigsqcap\{U \mid U \cup Y \geq Z\}$ exists.

**Proposition 1.** *The set $\mathcal{S}(X)$ of subattributes carries the structure of a lattice with zero and one and relative pseudo-complements, where the order $\geq$ is as defined in Definition 5, and $\lambda$ and $X$ are the zero and one.*

In the following we denote join by $\sqcup$, meet by $\sqcap$ and relative pseudo-complement by $\leftarrow$. Then it is straightforward to show the following properties:

- for the join $\sqcup$:

  1. $Y \sqcup Z = Y$ iff $Y \geq Z$;

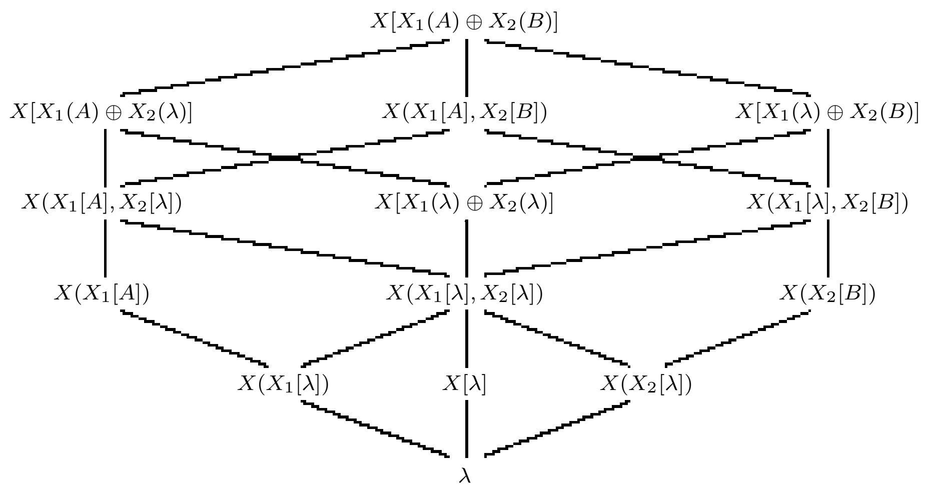
$X[X_1(A) \oplus X_2(B)]$
$X[X_1(A) \oplus X_2(\lambda)]$
$X(X_1[A], X_2[B])$
$X[X_1(\lambda) \oplus X_2(B)]$
$X(X_1[A], X_2[\lambda])$
$X[X_1(\lambda) \oplus X_2(\lambda)]$
$X(X_1[\lambda], X_2[B])$
$X(X_1[A])$
$X(X_1[\lambda], X_2[\lambda])$
$X(X_2[B])$
$X(X_1[\lambda])$
$X[\lambda]$
$X(X_2[\lambda])$
$\lambda$

Figure 1: The lattice $\mathcal{S}(X[X_1(A) \oplus X_2(B)])$

2. for $X = X(X_1, \dots, X_n)$, $Y = X(Y_1, \dots, Y_n)$ and $Z = X(Z_1, \dots, Z_n)$ we have $Y \sqcup Z = X(Y_1 \sqcup Z_1, \dots, Y_n \sqcup Z_n)$;
3. for $X = X(X_1, \dots, X_n)$, $Y = X(Y_1, \dots, Y_n) \neq \lambda$ and $Z = X_I[\lambda]$ with $I = \{i_1, \dots, i_k\}$ we have $Y \sqcup Z = Z \sqcup Y = Y \sqcup X(X_{i_1}[\lambda], \dots, X_{i_k}[\lambda])$;
4. for $X = X_1(X'_1) \oplus \cdots \oplus X_n(X'_n)$, $Y = X_1(Y'_1) \oplus \cdots \oplus X_n(Y'_n)$ and $Z = X_1(Z'_1) \oplus \cdots \oplus X_n(Z'_n)$ we have $Y \sqcup Z = X_1(Y'_1 \sqcup Z'_1) \oplus \cdots \oplus X_n(Y'_n \sqcup Z'_n)$;
5. for $X = X[X']$, $Y = X[Y']$ and $Z = X[Z']$ we have $Y \sqcup Z = X[Y' \sqcup Z']$;
6. for $X = X[X_1(X'_1) \oplus \cdots \oplus X_n(X'_n)]$, $Y = X[X_1(Y'_1) \oplus \cdots \oplus X_n(Y'_n)]$ and $Z = X(Z_1, \dots, Z_n)$ with either $Z_i = X_i[Z'_i]$ or $Z_i = \lambda$ we have $Y \sqcup Z = Z \sqcup Y = X[X_1(U_1) \oplus \cdots \oplus X_n(U_n)]$ with $U_i = \begin{cases} Y'_i \sqcup Z'_i & \text{for } Z_i = X_i[Z'_i] \\ Y'_i & \text{for } Z_i = \lambda \end{cases}$.

- for the meet $\sqcap$:

1. $Y \sqcap Z = Z$ iff $Y \geq Z$;
2. for $X = X(X_1, \dots, X_n)$, $Y = X(Y_1, \dots, Y_n)$ and $Z = X(Z_1, \dots, Z_n)$ we have $Y \sqcap Z = X(Y_1 \sqcap Z_1, \dots, Y_n \sqcap Z_n)$;
3. for $X = X(X_1, \dots, X_n)$, $Y = X(Y_1, \dots, Y_n) \neq \lambda$ and $Z = X_I[\lambda]$ with $I = \{i_1, \dots, i_k\}$ and $Y \not\geq Z$ we have $Y \sqcap Z = Z \sqcap Y = \lambda$;
4. for $X = X_1(X'_1) \oplus \cdots \oplus X_n(X'_n)$, $Y = X_1(Y'_1) \oplus \cdots \oplus X_n(Y'_n)$ and $Z = X_1(Z'_1) \oplus \cdots \oplus X_n(Z'_n)$ we have $Y \sqcap Z = X_1(Y'_1 \sqcap Z'_1) \oplus \cdots \oplus X_n(Y'_n \sqcap Z'_n)$;
5. for $X = X[X']$, $Y = X[Y']$ and $Z = X[Z']$ we have $Y \sqcap Z = X[Y' \sqcap Z']$;

6. for $X = X[X_1(X_1') \oplus \cdots \oplus X_n(X_n')]$, $Y = X[X_1(Y_1') \oplus \cdots \oplus X_n(Y_n')]$ and $Z = X(Z_1, \ldots, Z_n)$ with either $Z_i = X_i[Z_i']$ or $Z_i = \lambda$ we have $Y \sqcap Z = Z \sqcap Y = X(U_1, \ldots, U_n)$ with $U_i = \begin{cases} X_i[Y_i' \sqcap Z_i'] & \text{for } Z_i = X_i[Z_i'] \\ \lambda & \text{for } Z_i = \lambda \end{cases}$.

- for the relative pseudo-complement $\leftarrow$:

1. $\lambda \leftarrow Y = Y$;
2. for $Y \geq Z$ we have $Y \leftarrow Z = \lambda$;
3. for $X = X(X_1, \ldots, X_n)$, $Y = X(Y_1, \ldots, Y_n)$, $Z = X(Z_1, \ldots, Z_n)$ and $X[\lambda] \notin \mathcal{S}(X)$ we have $Y \leftarrow Z = X(Y_1 \leftarrow Z_1, \ldots, Y_n \leftarrow Z_n)$;
4. for $Z = X(Z_1, \ldots, Z_n) \neq \lambda$ and $I = \{i_1, \ldots, i_k\}$ we have $Z \leftarrow X_I[\lambda] = \lambda$ and $X_I[\lambda] \leftarrow Z = X(X_{i_1}[\lambda] \leftarrow Z_{i_1}, \ldots, X_{i_k}[\lambda] \leftarrow Z_{i_k})$;
5. for $X = X_1(X_1') \oplus \cdots \oplus X_n(X_n')$, $Y = X_1(Y_1) \oplus \cdots \oplus X_n(Y_n)$, $Z = X_1(Z_1) \oplus \cdots \oplus X_n(Z_n)$ and $Y \not\geq Z$ we have $Y \leftarrow Z = X_1(Y_1 \leftarrow Z_1) \oplus \cdots \oplus X_n(Y_n \leftarrow Z_n)$;
6. for $Z = X(Z_1, \ldots, Z_n) \neq \lambda$ we have $Z \leftarrow X[\lambda] = \lambda$ and $X[\lambda] \leftarrow Z = X(X_1[\lambda] \leftarrow Z_1, \ldots, X_n[\lambda] \leftarrow Z_n)$;
7. for $X = X[X_1(X_1') \oplus \cdots \oplus X_n(X_n')]$ or $X = X(X_1[X_1'], \ldots, X_n[X_n'])$ we have:
   (a) for $Z = X(Z_1, \ldots, Z_n) \neq \lambda$ and $I = \{i_1, \ldots, i_k\}$ we have $Z \leftarrow X_I[\lambda] = \lambda$ and $X_I[\lambda] \leftarrow Z = X(X_{i_1}[\lambda] \leftarrow Z_{i_1}, \ldots, X_{i_n}[\lambda] \leftarrow Z_{i_k})$;
   (b) for $Y = X(Y_1, \ldots, Y_n)$ and $Z = X(Z_1, \ldots, Z_n)$ with $\lambda \neq Y \not\geq Z$
      – if $Y_i \geq Z_i$ or $Y_i = \lambda, Z_i = X_i[\lambda]$ for all $i = 1, \ldots, n$ we have $Y \leftarrow Z = \lambda$,
      – otherwise we have $Y \leftarrow Z = X(Y_1 \leftarrow Z_1, \ldots, Y_n \leftarrow Z_n)$;
   (c) for $Y = X[X_1(Y_1') \oplus \cdots \oplus X_n(Y_n')]$ and $Z = X(Z_1, \ldots, Z_n)$ with $Z_i = X_i[Z_i']$ or $Z_i = \lambda$ and $Y \not\geq Z$ we have $Y \leftarrow Z = X(U_1, \ldots, U_n)$ with $U_i = \begin{cases} X_i[Y_i' \leftarrow Z_i'] & \text{for } Z_i \neq \lambda \\ \lambda & \text{else} \end{cases}$;
   (d) for $Y = X(Y_1, \ldots, Y_n) \neq \lambda$ with $Y_i = X_i[Y_i']$ or $Y_i = \lambda$, and $Z = X[X_1(Z_1') \oplus \cdots \oplus X_n(Z_n')]$ with $Z_i = X_i(Z_i')$ or $Z_i = \lambda$ and $Y \not\geq Z$ we have $Y \leftarrow Z = X[X_1(U_1) \oplus \cdots \oplus X_n(U_n)]$ with $U_i = \begin{cases} Y_i' \leftarrow Z_i' & \text{for } Y_i \neq \lambda \neq Z_i \\ \lambda & \text{else} \end{cases}$;
   (e) for $Y = X[X_1(Y_1') \oplus \cdots \oplus X_n(Y_n')]$, $Z = X[X_1(Z_1') \oplus \cdots \oplus X_n(Z_n')]$ with $Y \neq Z$ we have $Y \leftarrow Z = X(U_1, \ldots, U_n)$ with

$$U_i = \begin{cases} X_i[Y_i' \leftarrow Z_i'] & \text{for } Y_i' \neq \lambda \neq Z_i' \\ \lambda & \text{else} \end{cases}.$$

## 3 Axiomatisation of Functional Dependencies

In this section we will define functional dependencies on $\mathcal{S}(X)$ and derive some sound derivation rules. We consider finite sets $r \subseteq dom(X)$, which we will call simply *instances* of $X$. If $Y$ is a nested attribute that occurs inside $X$, then an instance $r$ of $X$ defines an instance $r(Y)$ of $Y$; simply take $r(Y) = \{v' \in dom(Y) \mid v'$ occurs inside some $v \in r$ at the position defined by $Y\}$.

**Definition 6.** Let $X, X' \in \mathcal{N}$ such that $X'$ occurs in $X$. A *functional dependency* (FD) on $\mathcal{S}(X)$ is an expression $X' : \mathcal{Y} \to \mathcal{Z}$ with $\mathcal{Y}, \mathcal{Z} \subseteq \mathcal{S}(X')$.

An instance $r$ of $X$ *satisfies the FD* $X' : \mathcal{Y} \to \mathcal{Z}$ on $\mathcal{S}(X)$ (notation: $r \models X' : \mathcal{Y} \to \mathcal{Z}$) iff for all $t_1, t_2 \in r(X')$ with $\pi_Y^{X'}(t_1) = \pi_Y^{X'}(t_2)$ for all $Y \in \mathcal{Y}$ we also have $\pi_Z^{X'}(t_1) = \pi_Z^{X'}(t_2)$ for all $Z \in \mathcal{Z}$.

Let $\Sigma$ be a set of FDs defined on some $\mathcal{S}(X)$. A FD $\psi$ is implied by $\Sigma$ (notation: $\Sigma \models \psi$) iff all instances $r$ with $r \models \varphi$ for all $\varphi \in \Sigma$ also satisfy $\psi$. As usual we write $\Sigma^* = \{\psi \mid \Sigma \models \psi\}$.

We write $\Sigma^+$ for the set of all FDs that can be derived from $\Sigma$ by applying a system $\mathfrak{R}$ of axioms and rules, i.e. $\Sigma^+ = \{\psi \mid \Sigma \vdash_{\mathfrak{R}} \psi\}$. We omit the standard definitions of derivations with a given rule system, and also write simply $\vdash$ instead of $\vdash_{\mathfrak{R}}$, if the rule system is clear from the context.

Our goal is to find a finite axiomatisation, i.e. a rule system $\mathfrak{R}$ such that $\Sigma^* = \Sigma^+$ holds. The rules in $\mathfrak{R}$ are *sound* iff $\Sigma^+ \subseteq \Sigma^*$ holds, and *complete* iff $\Sigma^* \subseteq \Sigma^+$ holds.

### 3.1 Sound Axioms and Rules for Functional Dependencies

Let us now look at derivation rules for FDs. We will need a particular notion of "semi-disjointness" that will permit a generalisation of the well known Armstrong axioms for the relational model.

**Definition 7.** Two subattributes $Y, Z \in \mathcal{S}(X)$ are called *semi-disjoint* iff one of the following holds:

1. $Y \geq Z$ or $Z \geq Y$;
2. $X = X(X_1, \dots, X_n)$, $Y = X(Y_1, \dots, Y_n)$, $Z = X(Z_1, \dots, Z_n)$ and $Y_i, Z_i \in \mathcal{S}(X_i)$ are semi-disjoint for all $i = 1, \dots, n$;
3. $X = X[X']$, $Y = X[Y']$, $Z = X[Z']$ and $Y', Z' \in \mathcal{S}(X')$ are semi-disjoint;
4. $X = X_1(X_1') \oplus \cdots \oplus X_n(X_n')$, $Y = X_1(Y_1') \oplus \cdots \oplus X_n(Y_n')$, $Z = X_1(Z_1') \oplus \cdots \oplus X_n(Z_n')$ and $Y_i', Z_i' \in \mathcal{S}(X_i')$ are semi-disjoint for all $i = 1, \dots, n$;
5. $X = X[X_1(X_1') \oplus \cdots \oplus X_n(X_n')]$, $Y = X(Y_1, \dots, Y_n)$ with $Y_i = X_i[Y_i']$ or $Y_i = \lambda = Y_i'$, $Z = X[X_1(Z_1') \oplus \cdots \oplus X_n(Z_n')]$, and $Y_i'$, $Z_i'$ are semi-disjoint for all $i = 1, \dots, n$.

With the notion of semi-disjointness we can formulate axioms and rules for FDs and show their soundness.

**Theorem 1.** *The following axioms and rules are sound for the implication of FDs:*

- *the $\lambda$ axiom: $X' : \emptyset \to \{\lambda\}$*
- *the subattribute axiom: $X' : \{Y\} \to \{Z\}$ for $Y \geq Z$*
- *the join axiom: $X' : \{Y, Z\} \to \{Y \sqcup Z\}$ for semi-disjoint $Y$ and $Z$*
- *the reflexivity axiom: $X' : \mathcal{Y} \to \mathcal{Z}$ for $\mathcal{Z} \subseteq \mathcal{Y}$*
- *the extension rule: $X' : \mathcal{Y} \to \mathcal{Z}$ implies $X' : \mathcal{Y} \to \mathcal{Y} \cup \mathcal{Z}$*
- *the transitivity rule: $X' : \mathcal{Y} \to \mathcal{Z}$ and $X' : \mathcal{Z} \to \mathcal{U}$ imply $X' : \mathcal{Y} \to \mathcal{U}$*
- *the list axioms:*
  - $X : \{X_I[\lambda], X_J[\lambda]\} \to \{X_{I\cup J}[\lambda]\}$ *for* $I \cap J = \emptyset$
  - $X : \{X_I[\lambda], X_{I\cup J}[\lambda]\} \to \{X_J[\lambda]\}$ *for* $I \cap J = \emptyset$
  - $X : \{X_I[\lambda], X_J[\lambda], X_{I\cap J}[\lambda]\} \to \{X_{(I-J)\cup(J-I)}[\lambda]\}$
  - $X : \{X_I[\lambda], X_J[\lambda], X_{(I-J)\cup(J-I)}[\lambda]\} \to \{X_{I\cap J}[\lambda]\}$
- *the list lifting rule: $X' : \mathcal{Y} \to \mathcal{Z}$ implies $X[X'] : \{X[Y] \mid Y \in \mathcal{Y}\} \to \{X[Z] \mid Z \in \mathcal{Z}\}$ for $\mathcal{Y} \neq \emptyset$*
- *the record lifting rule: $X_i : \mathcal{Y}_i \to \mathcal{Z}_i$ implies $X(X_1, \dots, X_n) : \bar{\mathcal{Y}_i} \to \bar{\mathcal{Z}_i}$ with $\bar{\mathcal{Y}_i} = \{X(\lambda, \dots, Y_i, \dots, \lambda) \mid Y_i \in \mathcal{Y}_i\}$ and $\bar{\mathcal{Z}_i} = \{X(\lambda, \dots, Z_i, \dots, \lambda) \mid Y_i \in \mathcal{Z}_i\}$*
- *the union lifting rule: $X_i' : \mathcal{Y}_i \to \mathcal{Z}_i$ implies $X_1(X_1') \oplus \cdots \oplus X_n(X_n') : \bar{\mathcal{Y}_i} \to \bar{\mathcal{Z}_i}$ with $\bar{\mathcal{Y}_i} = \{X_1(\lambda) \oplus \cdots \oplus X_i(Y_i) \oplus \cdots \oplus X_n(\lambda) \mid Y_i \in \mathcal{Y}_i\}$ and $\bar{\mathcal{Z}_i} = \{X_1(\lambda) \oplus \cdots \oplus X_i(Z_i) \oplus \cdots \oplus X_n(\lambda) \mid Z_i \in \mathcal{Z}_i\}$*

*Proof.* We only show the soundness of some of the axioms and rules. The proof for the other axioms and rules is either analogous or trivial.

For the join axion let $t_1, t_2 \in dom(X)$ with $\pi^X_Y(t_1) = \pi^X_Y(t_2)$ and $\pi^X_Z(t_1) = \pi^X_Z(t_2)$. We use induction on $X$ to show $\pi^X_{Y\sqcup Z}(t_1) = \pi^X_{Y\sqcup Z}(t_2)$. The cases $X = \lambda$ and $X = A$ (i.e. a simple attribute) are trivial. There is also nothing to show for $Y \geq Z$ or $Z \geq Y$, as in these cases $Y \sqcup Z$ is one of $Y$ or $Z$.

For $X = X(X_1, \dots, X_n)$ let $Y = X(Y_1, \dots, Y_n)$ and $Z = X(Z_1, \dots, Z_n)$ be semi-disjoint. For $t_j = (X_1 : t_{j1}, \dots, X_n : t_{jn})$ $(j = 1, 2)$ we have $\pi^{X_i}_{Y_i}(t_{1i}) = \pi^{X_i}_{Y_i}(t_{2i})$ and $\pi^{X_i}_{Z_i}(t_{1i}) = \pi^{X_i}_{Z_i}(t_{2i})$, and $Y_i$, $Z_i$ are semi-disjoint for all $i = 1, \dots, n$. By induction $\pi^{X_i}_{Y_i\sqcup Z_i}(t_{1i}) = \pi^{X_i}_{Y_i\sqcup Z_i}(t_{2i})$, which implies $\pi^X_{Y\sqcup Z}(t_1) = \pi^X_{Y\sqcup Z}(t_2)$.

For $X = X_1(X_1') \oplus \cdots \oplus X_n(X_n')$ assume $t_1 = (X_j : t_1')$ and $t_2 = (X_j : t_2')$. Thus, for semi-disjoint $Y = X_1(Y_1') \oplus \cdots \oplus X_n(Y_n')$ and $Z = X_1(Z_1') \oplus \cdots \oplus X_n(Z_n')$

we obtain $\pi^{X'_j}_{Y'_j}(t'_1) = \pi^{X'_j}_{Y'_j}(t'_2)$, $\pi^{X'_j}_{Z'_j}(t'_1) = \pi^{X'_j}_{Z'_j}(t'_2)$, and $Y'_j$, $Z'_j$ are semi-disjoint. By induction $\pi^{X'_j}_{Y'_j \sqcup Z'_j}(t'_1) = \pi^{X'_j}_{Y'_j \sqcup Z'_j}(t'_2)$, which implies $\pi^X_{Y \sqcup Z}(t_1) = \pi^X_{Y \sqcup Z}(t_2)$.

For $X = X[X_1(X'_1) \oplus \cdots \oplus X_n(X'_n)]$, $Y = X(Y_1, \dots, Y_n)$ with $Y_i = X_i[Y'_i]$ or $Y_i = \lambda = Y'_i$ and $Z = X[X_1(Z'_1) \oplus \cdots \oplus X_n(Z'_n)]$ we get $Y \sqcup Z = X[X_1(Y'_1 \sqcup Z'_1) \oplus \cdots \oplus X_n(Y'_n \sqcup Z'_n)]$. As $Z \geq X[\lambda]$, we also have $\pi^X_{X[\lambda]}(t_1) = \pi^X_{X[\lambda]}(t_2)$, so $t_1$ and $t_2$ are lists of equal length. Therefore, assume $t_j = [t_{j1}, \dots, t_{jm}]$ for $j = 1, 2$ and $t_{jk} = (X_\ell : t''_{jk})$. This gives $\pi^X_{Y \sqcup Z}(t_j) = [t'_{j1}, \dots, t'_{jm}]$ with $t'_{jk} = (X_\ell : \pi^{X'_\ell}_{Y'_\ell \sqcup Z'_\ell}(t''_{jk}))$. We know $\pi^{X'_\ell}_{Z'_\ell}(t''_{1k}) = \pi^{X'_\ell}_{Z'_\ell}(t''_{2k})$, so we are done for $Y_\ell = \lambda$. For $Y_\ell \neq \lambda$ the sublists containing all $(X_\ell : t''_{jk})$ coincide on $Y'_\ell$. As $Y'_\ell$ and $Z'_\ell$ are semi-disjoint, we have $\pi^{X'_\ell}_{Y'_\ell \sqcup Z'_\ell}(t''_{1k}) = \pi^{X'_\ell}_{Y'_\ell \sqcup Z'_\ell}(t''_{2k})$ by induction, which implies $\pi^X_{Y \sqcup Z}(t_1) = \pi^X_{Y \sqcup Z}(t_2)$.

For the first list axiom let $t_1, t_2 \in dom(X)$. Then $\pi^X_{X_I[\lambda]}(t_1) = \pi^X_{X_I[\lambda]}(t_2)$ means that $t_1$ and $t_2$ contain the same number of elements of the form $(X_i : v_i)$ with $i \in I$. If the same holds for $I \cup J$, then $t_1$ and $t_2$ must also contain the same number of elements of the form $(X_i : v_i)$ with $i \in J$, i.e. $\pi^X_{X_J[\lambda]}(t_1) = \pi^X_{X_J[\lambda]}(t_2)$. The soundness of the second list axiom follows from the same argument.

Analogously, for the third list axiom $\pi^X_Y(t_1) = \pi^X_Y(t_2)$ for $Y \in \{X_I[\lambda], X_J[\lambda], X_{I \cap J}[\lambda]\}$ means that $t_1$, $t_2$ contain the same number of elements with labels in $I$, $J$ and $I \cap J$, respectively. So they also contain the same number of elements with labels in $(I - J) \cup (J - I)$. The soundness of the fourth list axiom follows from the same argument.

For the soundness of the list lifting rule let $t_1, t_2 \in dom(X)$ with $\pi^X_{X[Y]}(t_1) = \pi^X_{X[Y]}(t_2)$ for all $X[Y]$ with $Y \in \mathcal{Y}$. As $\mathcal{Y} \neq \emptyset$, it follows that $t_1$ and $t_2$ must have the same length, say $t_i = [t_{i1}, \dots, t_{ik}]$ $(i = 1, 2)$, and for all $j = 1, \dots, k$ and all $Y \in \mathcal{Y}$ we have $\pi^{X'}_Y(t_{1j}) = \pi^{X'}_Y(t_{2j})$. Hence $\pi^{X'}_Z(t_{1j}) = \pi^{X'}_Z(t_{2j})$ for all $j = 1, \dots, k$ and all $Z \in \mathcal{Z}$, which implies $\pi^X_{X[Z]}(t_1) = \pi^X_{X[Z]}(t_2)$ for all $X[Z]$ with $Z \in \mathcal{Z}$. The soundness of the other two lifting rules follows analogously. $\square$

Using these rules we can derive additional rules:

- the union rule: $X : \mathcal{Y} \to \mathcal{Z}$ and $X : \mathcal{Y} \to \mathcal{U}$ imply $X : \mathcal{Y} \to \mathcal{Z} \cup \mathcal{U}$
- the fragmentation rule: $X : \mathcal{Y} \to \mathcal{Z}$ implies $X : \mathcal{Y} \to \{Z\}$ for $Z \in \mathcal{Z}$
- the join rule: $X : \{Y\} \to \{Z\}$ implies $X : \{Y\} \to \{Y \sqcup Z\}$ for semi-disjoint $Y$ and $Z$

## 3.2 SHL-Ideals

In this subsection we investigate ideals. Of particular interest will be ideals with additional closure properties, which we call "strong high-level ideals" or SHL-ideals for short. These ideals will appear naturally in the completeness proof in the next subsection. The main result of this subsection is Theorem 2.

**Definition 8.** An *ideal* for a nested attribute $X$ is a subset $\mathcal{G} \subseteq \mathcal{S}(X)$ with $\lambda \in \mathcal{G}$ and whenever $Y \in \mathcal{G}$, $Z \in \mathcal{S}(X)$ with $Y \geq Z$, then also $Z \in \mathcal{G}$.

Let us now address the closure properties that will turn ideals into "higher-level" or "strong higher-level ideals".

**Definition 9.** Let $X \in \mathcal{N}$. An ideal $\mathcal{F} \subseteq \mathcal{S}(X)$ is called *SHL-ideal* on $\mathcal{S}(X)$ iff the following properties are satisfied:

1. if $Y, Z \in \mathcal{F}$ are semi-disjoint, then $Y \sqcup Z \in \mathcal{F}$;
2. (a) if $X_I[\lambda] \in \mathcal{F}$ and $X_J[\lambda] \in \mathcal{F}$ with $I \subseteq J$, then $X_{J-I}[\lambda] \in \mathcal{F}$;
   (b) if $X_I[\lambda] \in \mathcal{F}$ and $X_J[\lambda] \in \mathcal{F}$ with $I \cap J = \emptyset$, then $X_{I \cup J}[\lambda] \in \mathcal{F}$;
   (c) if $X_I[\lambda] \in \mathcal{F}$ and $X_J[\lambda] \in \mathcal{F}$, then $X_{I \cap J}[\lambda] \in \mathcal{F}$ iff $X_{(I-J)\cup(J-I)}[\lambda] \in \mathcal{F}$;
3. if $X = X(X'_1, \ldots, X'_n)$, then the sets $\mathcal{F}_i = \{Y_i \in \mathcal{S}(X'_i) \mid X(\lambda, \ldots, Y_i, \ldots, \lambda) \in \mathcal{F}\}$ are SHL-ideals;
4. if $X = X[X']$ and $\mathcal{F} \neq \{\lambda\}$, then the set $\mathcal{G} = \{Y \in \mathcal{S}(X') \mid X[Y] \in \mathcal{F}\}$ is a SHL-ideal;
5. If $X = X_1(X'_1) \oplus \cdots \oplus X_n(X'_n)$ and $\mathcal{F} \neq \{\lambda\}$, then the sets $\mathcal{F}_i = \{Y_i \in \mathcal{S}(X'_i) \mid X_1(\lambda) \oplus \cdots \oplus X_i(Y_i) \oplus \cdots \oplus X_n(\lambda) \in \mathcal{F}\}$ are SHL-ideals.

We now prove the main result of this subsection.

**Theorem 2.** *Let $X$ be a nested attribute such that the union-constructor appears in $X$ only inside a list-constructor. If $\mathcal{F}$ is a SHL-ideal on $\mathcal{S}(X)$, then there exist tuples $t_0, t_1 \in dom(X)$ with $\pi^X_Y(t_0) = \pi^X_Y(t_1)$ iff $Y \in \mathcal{F}$.*

*Proof.* We use induction on $X$. The case $X = \lambda$ is trivial. For a simple attribute $X = A$ we either have $\mathcal{F} = \{\lambda\}$ or $\mathcal{F} = \{A, \lambda\}$. In the former case take $t_0 = a$ and $t_1 = a'$ for $a, a' \in dom(A)$ with $a \neq a'$. In the latter case take $t_0 = t_1 = a$.

Let $X = X(X_1, \ldots, X_n)$ with $X[\lambda] \notin \mathcal{S}(X)$. Take the SHL-ideals $\mathcal{F}_i$ from Definition 9(3). By induction we find $t_{0i}, t_{1i} \in dom(X_i)$ with $\pi^{X_i}_{Y_i}(t_{0i}) = \pi^{X_i}_{Y_i}(t_{1i})$ iff $Y_i \in \mathcal{F}_i$. So take $t_j = (X_1 : t_{j1}, \ldots, X_n : t_{jn})$ $(j = 0, 1)$. For $Y = X(Y_1, \ldots, Y_n) \in \mathcal{F}$ we have $\pi^X_Y(t_0) = (X_1 : \pi^{X_1}_{Y_1}(t_{01}), \ldots, X_n : \pi^{X_n}_{Y_n}(t_{0n})) = (X_1 : \pi^{X_1}_{Y_1}(t_{11}), \ldots, X_n : \pi^{X_n}_{Y_n}(t_{1n})) = \pi^X_Y(t_1)$. For $Y = X(Y_1, \ldots, Y_n) \notin \mathcal{F}$ there is at least one $Y_i \notin \mathcal{F}_i$, which gives $\pi^X_Y(t_0) = (X_1 : \pi^{X_1}_{Y_1}(t_{01}), \ldots, X_n : \pi^{X_n}_{Y_n}(t_{0n})) \neq (X_1 : \pi^{X_1}_{Y_1}(t_{11}), \ldots, X_n : \pi^{X_n}_{Y_n}(t_{1n})) = \pi^X_Y(t_1)$.

Let $X = X[X']$ and assume that $X'$ is not a union attribute. If we have $\mathcal{F} = \{\lambda\}$, then take $t_0 = [v]$ with $v \in dom(X')$ and $t_1 = []$. For $Y = X[Y'] \notin \mathcal{F}$ we get $\pi^X_Y(t_0) = [\pi^{X'}_{Y'}(v)] \neq [] = \pi^X_Y(t_1)$. For $\mathcal{F} \neq \{\lambda\}$ take the SHL-ideal $\mathcal{G}$ from Definition 9(4). By induction we find $t'_0, t'_1 \in dom(X')$ with $\pi^{X'}_{Y'}(t'_0) = \pi^{X'}_{Y'}(t'_1)$ iff $Y' \in \mathcal{G}$. Let $t_j = [t'_j]$ for $j = 0, 1$. Then we get $\pi^X_Y(t_0) = [\pi^{X'}_{Y'}(t'_0)] = [\pi^{X'}_{Y'}(t'_1)] = \pi^X_Y(t_1)$ iff $Y = X[Y'] \in \mathcal{F}$.

Let $X = X[X_1(X_1') \oplus \cdots \oplus X_n(X_n')]$. If $\mathcal{F} = \{\lambda\}$, define $t_0 = [(X_1 : v_1), \dots, (X_n : v_n)]$ with arbitrary $v_i \in dom(X_i')$ and $t_1 = []$. Then we get $\pi^X_{X_I\{\lambda\}}(t_0) = [\underbrace{\top, \dots, \top}_{|I| \text{ times}}]$, whereas $\pi^X_{X_I\{\lambda\}}(t_1) = []$.

Now assume $\mathcal{F} \neq \{\lambda\}$. Take $I^+ = \{i \in \{1, \dots, n\} \mid X(X_i[\lambda]) \in \mathcal{F}\}$ and $I^- = \{1, \dots, n\} - I^+$. If $I^+ = \{i_1, \dots, i_k\}$, then consider first the subattribute $X^+ = X[X_{i_1}(X'_{i_1}) \oplus \cdots \oplus X_{i_k}(X'_{i_k})]$. By Definition 9(2b) we have $X_I[\lambda] \in \mathcal{F}$ for all $I \subseteq I^+$. We first construct $t_0^+, t_1^+ \in dom(X^+)$ with $\pi^{X^+}_Y(t_0^+) = \pi^{X^+}_Y(t_1^+)$ iff $Y \in \mathcal{F}^+ = \{Y \in \mathcal{F} \mid X^+ \geq Y\}$.

For this take $\tilde{X} = X(X_{i_1}[X'_{i_1}], \dots, X_{i_k}[X'_{i_k}])$ and $\mathcal{H} = \{Y = X(Y_{i_1}, \dots, Y_{i_k}) \mid Y \in \mathcal{F}\}$. Ignoring restructuring and considering $\tilde{X}$ just as a record attribute, $\mathcal{H}$ becomes an SHL-ideal on $\mathcal{S}(\tilde{X})$. Applying the record case above we obtain $\tilde{t}_0, \tilde{t}_1 \in dom(\tilde{X})$ with $\pi^{\tilde{X}}_Y(\tilde{t}_0) = \pi^{\tilde{X}}_Y(\tilde{t}_1)$ iff $Y \in \mathcal{H}$.

If $\tilde{t}_i = (t_{i,i_1}, \dots, t_{i,i_k})$, we may concatenate these lists in the order of the indices to define $t_0^+$ and $t_1^+$, respectively. Then for $Y \in \mathcal{S}(X^+)$ with $\tilde{X} \geq Y$ we have $\pi^{X^+}_Y(t_0^+) = \pi^{X^+}_Y(t_1^+)$ iff $Y \in \mathcal{F}^+$. This does not change, if for any $j$ we replace $t_{0,i_j}$ and $t_{1,i_j}$ by the concatenated lists $t_{0,i_j}{}^\frown t_{0,i_j}$ and $t_{1,i_j}{}^\frown t_{0,i_j}$, respectively.

Now let $K = \{k_1, \dots, k_m\} \subseteq I^+$ be maximal such that $X[X_{k_1}(X'_{k_1}) \oplus \cdots \oplus X_{k_m}(X'_{k_m})] \in \mathcal{F}$. Then for $k \in I^+ - K$ we must have $X(X_k[X'_k]) \notin \mathcal{F}$, otherwise also $X[X_{k_1}(X'_{k_1}) \oplus \cdots \oplus X_{k_m}(X'_{k_m}) \oplus X_k(X'_k)] \in \mathcal{F}$ due to the semi-disjointness of the two subattributes and property 1 in Definition 9. Therefore, $K$ is uniquely determined.

Now, if $X(X_{i_1}[Y'_{i_1}], \dots, X_{i_\mu}[Y'_{i_\mu}]) \in \mathcal{F}$, but $X[X_{i_1}(Y'_{i_1}) \oplus \cdots \oplus X_{i_\mu}(Y'_{i_\mu})] \notin \mathcal{F}$, then the uniqueness of $K$ implies $X(X_{i_1}[X'_{i_1}], \dots, X_{i_\mu}[X'_{i_\mu}]) \notin \mathcal{F}$. Hence there must be some $\iota \in \{i_1, \dots, i_\mu\}$ with $t_{0,\iota} \neq t_{1,\iota}$. We therefore replace $t_{0,\iota}$ and $t_{1,\iota}$ by the concatenated lists $t_{0,\iota}{}^\frown t_{0,\iota}$ and $t_{1,\iota}{}^\frown t_{0,\iota}$, respectively, changing $t_0^+$ and $t_1^+$ accordingly. This gives $\pi^{X^+}_{X[X_{i_1}(Y'_{i_1}) \oplus \cdots \oplus X_{i_\mu}(Y'_{i_\mu})]}(t_0^+) \neq \pi^{X^+}_{X[X_{i_1}(Y'_{i_1}) \oplus \cdots \oplus X_{i_\mu}(Y'_{i_\mu})]}(t_1^+)$ without destroying previously established equalities and inequalities. This implies $\pi^{X^+}_Y(t_0^+) = \pi^{X^+}_Y(t_1^+)$ iff $Y \in \mathcal{F}^+$ for all $Y \in \mathcal{S}(X^+)$ as claimed.

Now let $I^- = \{j_1, \dots, j_\ell\}$. We choose non-negative integers $x_i$, $y_i$ $(i = 1, \dots, \ell)$ such that for each $I = \{j_{r_1}, \dots, j_{r_{|I|}}\} \subseteq I^-$ we have

$$\sum_{p=1}^{|I|} x_{r_p} = \sum_{p=1}^{|I|} y_{r_p} \text{ iff } X_I[\lambda] \in \mathcal{F}.$$

These integers can be obtained by the following procedure:

```
for $p = 1, \dots, \ell$ :
    choose $x_p$, $y_p$ such that all equations and inequations containing
    only $x_i$, $y_i$ with $1 \leq i \leq p$ are satisfied;
    replace $x_p$, $y_p$ in the remaining equations and inequations by the
    chosen values
endfor
```

Properties 2(b) and 2(c) in Definition 9 guarantee that this procedure always produces a solution for the given equations and inequations. Now define

$$t_0^- = [\underbrace{(X_{j_1} : v_{j_1})}_{x_{j_1}\text{-times}}, \dots, \underbrace{(X_{j_\ell} : v_{j_\ell})}_{x_{j_\ell}\text{-times}}] \qquad \text{and} \qquad t_1^- = [\underbrace{(X_{j_1} : v_{j_1})}_{y_{j_1}\text{-times}}, \dots, \underbrace{(X_{j_\ell} : v_{j_\ell})}_{y_{j_\ell}\text{-times}}]$$

with arbitrary values $v_{j_i} \in dom(X'_{j_i})$ and concatenate $t_i^+$ and $t_i^-$ to give $t_i$ $(i = 0, 1)$. Then for $Y \in \{Y \in \mathcal{S}(X) \mid X^+ \geq Y\}$ we have $\pi_Y^X(t_i) = \pi_Y^X(t_i^+)$, hence $\pi_Y^X(t_0) = \pi_Y^X(t_1)$ iff $Y \in \mathcal{F}^+$.

For $Y \not\leq X^+$ we always have one $j \in I^-$ with $Y \geq X(X_j[\lambda])$ or $Y = X_I[\lambda]$. In the first case $Y \notin \mathcal{F}$ and

$$\pi^X_{X(X_j[\lambda])}(t_0) = \pi^X_{X(X_j[\lambda])}(t_0^-) \neq \pi^X_{X(X_j[\lambda])}(t_1^-) = \pi^X_{X(X_j[\lambda])}(t_1)$$

as desired. In the second case $\pi_Y^X(t_0) = \pi_Y^X(t_1)$ iff $\pi^X_{X_{I\cap I^-}[\lambda]}(t_0^-) = \pi^X_{X_{I\cap I^-}[\lambda]}(t_1^-)$ iff $X_{I\cap I^-}[\lambda] \in \mathcal{F}$ iff $Y = X_I[\lambda] \in \mathcal{F}$ due to property 2(a) of Definition 9 and $X_{I\cap I^+}[\lambda] \in \mathcal{F}$. □

## 3.3 Completeness of the Axioms and Rules for Functional Dependencies

In this final subsection we want to show that the axioms and rules from Theorem 1 are also complete. This gives our main result.

Before we come to the proof let us make a little observation on the union-constructor. If $X = X_1(X'_1) \oplus \cdots \oplus X_n(X'_n)$, then each instance $r$ of $X$ can be partitioned into $r_i$ $(i = 1, \dots, n)$, where $r_i$ contains exactly the $X_i$-labelled elements of $r$. Then $r$ satisfies a FD $\varphi \equiv \mathcal{Y} \rightarrow \mathcal{Z}$ iff each $r_i$ satisfies the $i$'th projection $\varphi_i$ of $\varphi$, which results by replacing all subattributes $Y = X_1(Y_1) \oplus \cdots \oplus X_n(Y_n)$ in $\mathcal{Y}$ or $\mathcal{Z}$ by $X_i(Y_i)$. Similarly, we see $\varphi \in \Sigma^+$ iff $\varphi_i \in \Sigma_i^+$ for all $i = 1, \dots, n$.

**Theorem 3.** *The axioms and rules in Theorem 1 are complete for the implication of FDs.*

*Proof.* Assume $\mathcal{Y} \rightarrow \mathcal{Z} \notin \Sigma^+$. Due to the union rule we must have $\mathcal{Y} \rightarrow \{Z\} \notin \Sigma^+$ for some $Z \in \mathcal{Z}$. Now take $\bar{\mathcal{Y}} = \{Z \mid \mathcal{Y} \rightarrow \{Z\} \in \Sigma^+\}$, so $Z \notin \bar{\mathcal{Y}}$. It is easy to see that $\mathcal{F} = \bar{\mathcal{Y}}$ is a SHL-ideal:

1. $\lambda \in \bar{\mathcal{Y}}$ follows from the reflexivity axiom, the subattribute axiom and the transitivity rule.
2. In the same way for $Z \in \bar{\mathcal{Y}}$ and $Z \geq Z'$ we get $Z' \in \bar{\mathcal{Y}}$ from the subattribute axiom and the transitivity rule.
3. For semi-disjoint $Z, Z' \in \bar{\mathcal{Y}}$ we obtain $Z \sqcup Z' \in \bar{\mathcal{Y}}$ from the union rule, the join axiom and the transitivity rule.

4. The other properties of SHL-ideals follow directly from the list axioms and the lifting rules.

If the outermost constructor is not the union-constructor, we can apply Theorem 2 to obtain an instance $r = \{t_1, t_2\}$ with $\pi^X_Z(t_1) = \pi^X_Z(t_2)$ iff $Z \in \bar{\mathcal{Y}}$. As $\mathcal{Y} \subseteq \bar{\mathcal{Y}}$ and $Z \notin \bar{\mathcal{Y}}$, we must have $r \not\models \mathcal{Y} \to \{Z\}$ and thus also $r \not\models \mathcal{Y} \to \mathcal{Z}$ due to the soundness of the fragmentation rule.

If the outermost constructor is the union-constructor, say $X = X_1(X'_1) \oplus \cdots \oplus X_n(X'_n)$ and thus $Z = X_1(Z_1) \oplus \cdots \oplus X_n(Z_n)$, we find some $i$ with $Z_i \notin \mathcal{F}_i$. Otherwise all $X_1(\lambda) \oplus \cdots \oplus X_i(Z_i) \oplus \cdots \oplus X_n(\lambda) \in \mathcal{F}$, and as these attributes are all semi-disjoint, we would obtain $Z \in \mathcal{F}$, too, which contradicts our assumptions.

Apply the Theorem 2 to $X'_i$ and $\mathcal{F}_i$, which gives $t_{i1}, t_{i2} \in \mathit{dom}(X'_i)$ with $\pi^{X'_i}_{Y_i}(t_{i1}) = \pi^{X'_i}_{Y_i}(t_{i2})$ iff $Y_i \in \mathcal{F}_i$. Take $r = \{(X_i : t_{i1}), (X_i : t_{i2})\}$. For $Y \in \mathcal{Y}$ we get

$$\pi^X_Y((X_i : t_{i1})) = (X_i : \pi^{X'_i}_{Y_i}(t_{i1})) = (X_i : \pi^{X'_i}_{Y_i}(t_{i2})) = \pi^X_Y((X_i : t_{i2}))$$

and on the other hand

$$\pi^X_Z((X_i : t_{i1})) = (X_i : \pi^{X'_i}_{Z_i}(t_{i1})) \neq (X_i : \pi^{X'_i}_{Z_i}(t_{i2})) = \pi^X_Z((X_i : t_{i2}))\ ,$$

i.e. $r \not\models \mathcal{Y} \to \{Z\}$ and thus also $r \not\models \mathcal{Y} \to \mathcal{Z}$ follows also in this case.

In order to complete the proof we have to show $r \models \Sigma$. Let $X' : \mathcal{V} \to \mathcal{W} \in \Sigma$. Applying only the lifting rules we obtain $X : \mathcal{V}^* \to \mathcal{W}^* \in \Sigma^+$. We either have $\mathcal{V}^* \subseteq \bar{\mathcal{Y}}$ or not. In the first case we obtain $\mathcal{Y} \to \mathcal{V}^* \in \Sigma^+$ and thus also $\mathcal{Y} \to \mathcal{W}^* \in \Sigma^+$, which implies $\mathcal{W}^* \subseteq \bar{\mathcal{Y}}$. This gives $\pi^X_W(t_1) = \pi^X_W(t_2)$ for all $W \in \mathcal{W}^*$ and hence $r(X') \models \mathcal{V} \to \mathcal{W}$. If $\mathcal{V}^* \not\subseteq \bar{\mathcal{Y}}$, then there is some $V \in \mathcal{V}^* - \bar{\mathcal{Y}}$, for which we must have $\pi^X_V(t_1) \neq \pi^X_V(t_2)$. This implies also $r(X') \models \mathcal{V} \to \mathcal{W}$. □

## 4 Conclusion

In this article we extended our theory of functional dependencies for higher-order data models and presented an axiomatisation for functional dependencies over XML documents. The approach is based on a representation of XML document type definitions (or XML schemata) by nested attributes using constructors for records, disjoint unions and lists, and a particular null value, which covers optionality. The list-constructor is used to represent the Kleene-star in regular expressions in XML element definitions.

In order to fully capture functional dependencies over XML documents we extended our previous work in two major directions. We introduced rational tree attributes, which result from reference structures in XML documents. This led to infinite subattribute lattices, but did not affect the derivation of dependencies. This is the first time that the investigation of functional dependencies for XML did not neglect references. Furthermore, we considered functional dependencies on embedded elements. These dependencies can be lifted through the constructors, i.e. they induce functional dependencies on complete XML documents. Such dependencies have not been considered in previous work.

Using a partial order on nested attributes we obtain the structure of non-distributive Brouwer algebras. The operations of the Brouwer algebra are exploited in the soundness and completeness proofs for derivation rules for functional dependencies.

In our theory it is also possible to use the multiset- or set-constructor instead of the list-constructor, thus neglecting order or duplicates. We may also treat all three "bulk" constructors together. These extensions had to be left out in this article. A glimpse of the necessary extensions for the other two bulk constructors can be obtained from [10].

Natural next steps in the development of a fully satisfying dependency theory for XML will be the generalisation to other classes of dependencies, e.g. multi-valued or join dependencies, the investigation of efficient closure algorithms, and the study of normal forms [15] that provably characterise desirable properties of well-designed XML documents. First steps in this direction are the normal forms introduced in [3], [8], and [17].

## References

[1] Abiteboul, S., Buneman, P., and Suciu, D. *Data on the Web: From Relations to Semistructured Data and XML.* Morgan Kaufmann Publishers, 2000.

[2] Abiteboul, S., and Hull, R. Restructuring hierarchical database objects. *Theoretical Computer Science* (1988).

[3] Arenas, M., and Libkin, L. A normal form for XML documents. In *PODS 2002* (2002), ACM.

[4] Buneman, P., Davidson, S., Fan, W., Hara, C., and Tan, W. Keys for XML. In *Tenth WWW Conference* (2001), IEEE.

[5] Goldfarb, C., and Prescod, P. *The XML Handbook.* Prentice Hall, 1998.

[6] Hartmann, S., Hoffmann, A., Link, S., and Schewe, K.-D. Axiomatizing functional dependencies in the higher-order entity relationship model. *Information Processing Letters 87*, 3 (2003), 133–137.

[7] Hartmann, S., and Link, S. Reasoning about functional dependencies in an abstract data model. *Electronic Notes in Theoretical Computer Science 84* (2003).

[8] Hartmann, S., Link, S., and Schewe, K.-D. A new normal form for conceptual databases. In *Information Modelling and Knowledge Bases XV* (2004), Y. Kiyoki, E. Kawaguchi, H. Jaakkola, and H. Kangassalo, Eds., vol. 105 of *Frontiers in Artificial Intelligence and Applications*, IOS Press, pp. 88–105.

[9] HARTMANN, S., LINK, S., AND SCHEWE, K.-D. Reasoning about functional and multi-valued dependencies in the presence of lists. In *Foundations of Information and Knowledge Systems* (2004), D. Seipel and J. M. Turull Torres, Eds., vol. 2942 of *Springer LNCS*, Springer Verlag.

[10] HARTMANN, S., LINK, S., AND SCHEWE, K.-D. Weak functional dependencies in higher-order datamodels. In *Foundations of Information and Knowledge Systems* (2004), D. Seipel and J. M. Turull Torres, Eds., vol. 2942 of *Springer LNCS*, Springer Verlag.

[11] SALI, A. Minimal keys in higher-order datamodels. In *Foundations of Information and Knowledge Systems* (2004), D. Seipel and J. M. Turull Torres, Eds., vol. 2942 of *Springer LNCS*, Springer Verlag.

[12] SCHEWE, K.-D., AND THALHEIM, B. Fundamental concepts of object oriented databases. *Acta Cybernetica 11*, 4 (1993), 49–85.

[13] THALHEIM, B. *Dependencies in Relational Databases.* Teubner-Verlag, 1991.

[14] THALHEIM, B. *Entity-Relationship Modeling: Foundations of Database Technology.* Springer-Verlag, 2000.

[15] VINCENT, M. *The semantic justification for normal forms in relational database design.* PhD thesis, Monash University, Melbourne, Australia, 1994.

[16] VINCENT, M. W., AND LIU, J. Functional dependencies for XML. In *Web Technologies and Applications: 5th Asia-Pacific Web Conference* (2003), vol. 2642 of *LNCS*, Springer-Verlag, pp. 22–34.

[17] VINCENT, M. W., AND LIU, J. Multivalued dependencies and a 4nf for XML. In *Advanced Information Systems Engineering: 15th International Conference CAiSE 2003* (2003), vol. 2681 of *LNCS*, Springer-Verlag, pp. 14–29.

[18] VINCENT, M. W., AND LIU, J. Multivalued dependencies in XML. In *British National Conference on Database Systems: BNCOD 2003* (2003), vol. 2712 of *LNCS*, Springer-Verlag, pp. 4–18.

*Received May, 2004*