Redundancy, Dependencies and Normal Forms for XML Databases

Klaus-Dieter Schewe

Massey University, Department of Information Systems
& Information Science Research Centre
Private Bag 11 222, Palmerston North, New Zealand,
Email: k.d.schewe@massey.ac.nz

Abstract

With the advent of XML and its use as a database language, dependency and normal form theory has attracted novel research interest. Several approaches to build up a dependency and normal form theory for XML databases have been published, mainly concentrating on functional dependencies and keys. XML-like database structures can be modelled by rational trees using constructors for lists and disjoint unions. This involves restructuring rules on subattributes. The absence of redundancy can be characterised by the nested list normal form. If ordering is ignored, constructors for sets or multisets have to be employed. For these the theory can be extended using counter-free functional dependencies. Finally, for keys an important research question is which systems of subattributes permit Armstrong instances. While this gives just a glimpse of a starting promising theory, a research agenda for further research will be set up.

Keywords: eXtensible Markup Language, functional dependencies, redundancy, normal forms, type constructors

1 XML and Databases

With the increase of data-intensive web applications the eXtensible Markup Language (XML) (Abiteboul, Buneman & Suciu 2000, Goldfarb & Prescod 1998) has conquered the field of databases. It is argued that XML can be used as a database language, which would not only support the data exchange on the web. This has led to significant research efforts considering

- the storage of XML documents in relational databases (Bohannon, Freire, Roy & Simeon 2002, Deutsch, Fernandez & Suciu 1999, Florescu & Kossmann 1999, Schmidt, Kersten, Windhouwer & Waas 2000, Shanmugasundaram, Tufte, Zhang, He, De Witt & Naughton 1999);
- query languages for XML (Abiteboul, Quass, McHugh, Widom & Wiener 1997, Buneman, Davidson, Hillebrand & Suciu 1996, Deutsch, Fernandez, Florescu, Levy & Suciu 1999), which by now has led to the de faxto standard of XQuery (World Wide Web Consortium (W3C) 2002);
- schema languages for XML (Buneman, Davidson, Fernandez & Suciu 1997, Cluet, Delobel, Siméon & Smaga 1998), which by now has led to the widely accepted XML Schema (World Wide Web Consortium (W3C) 2001);
- updates of XML documents (Tatarinov, Ives, Halevy & Weld 2001);
- and dependency and normal form theory (Arenas & Libkin 2004, Buneman, Davidson, Fan, Hara & Tan 2001, Fan & Libkin 2001, Vincent, Liu & Liu 2004).

However, the use of XML for databases is only one fact of this language. Other applications, say simply annotating text by comments on the content or cross-referencing, requires other features from XML, and the language supports them all. Thus, XML will never be an ideal database language. What is really interesting, is that XML databases just stand for databases that store trees, to be precise rational trees. We will take up this view here and therefore do not spend too much effort on syntax.

The focus of this paper is on dependency and normal form theory. This theory concerns the old question of well-designed databases or in other words the syntactic characterisation of semantically desirable properties. These properties are tightly connected with dependencies such as keys, functional dependencies, weak functional dependencies, equality-generating dependencies, multi-valued dependencies, inclusion dependencies, join dependencies, etc. All these classes of dependencies (and many more) have been deeply investigated in the context of the relational datamodel (Paredaens, De Bra, Gyssens & Van Gucht 1989, Thalheim 1991). The work now requires its generalisation to (rational) trees.

It seems there is an agreement that starting with functional dependencies (FDs) is a good idea. Abstractly speaking, a FD in the RDM expresses a constraint that whenever certain projections are equal, other projections must be equal as well. Then the question, which projections to consider was easy for the RDM. There are only sets of attributes, so we have to consider projections of tuples to attributes. In XML-like datamodels, i.e. models dealing with trees, the decision which projections to base the definition of FDs on has already led to different kinds of FDs without yet a common agreement. For instance, (Arenas & Libkin 2002, Arenas & Libkin 2004) consider paths in the tree based on the schema information, i.e. a DTD in their approach. The work in (Vincent & Liu 2003*a*, Vincent et al. 2004) also uses paths, but not DTDs. The work in (Hartmann & Link 2003) tries to extend these approaches considering subtrees instead of paths.

Alternatively, we may construct trees – even rational trees – using constructors for records, lists and unions, or for sets or multisets, if the order that comes with lists is considered non-desirable. This is the approach taken in (Hartmann, Link & Schewe 2004*a*,

Copyright ©2005, Australian Computer Society, Inc. This paper appeared at the Sixteenth Australasian Database Conference (ADC2005), University of Newcastle, Newcastle, Australia. Conferences in Research and Practice in Information Technology, Vol. 39. Gill Dobbie and Hugh Williams, Eds. Reproduction for academic, not-for profit purposes permitted provided this text is included.

Hartmann, Link & Schewe 2004*b*, Hartmann, Link & Schewe 2004*d*). In fact, this is also the approach taken by XML Schema. However, it is not sufficient to use the approach that has been developed for nested relations (Mok, Ng & Embley 1996, Özsoyoglû & Yuan 1987), as FDs have only been defined for unnested structures, which cannot capture all kinds of FDs.

In Section 2 we present a formal model for rational trees focussing on constructors for lists and disjoint unions. These form the basis for an axiomatisation of functional dependencies in Section 3. We continue this view loking at a normal form in Section 4, which allows us to characterise non-redundancy. In Section 5 we widen the scope looking not only at lists, but also at sets and multisets. Strictly speaking this leaves the grounds of XML, but it is perfectly fine with trees. We present an axiomatisation for a restricted class of functional dependencies. In Section 6 we extend this discussion looking at systems of minimal keys and Armstrong instances. Finally, in Section 7 we look at open research problems. In fact, there are much more open problems in this area than solutions so far.

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. We follow (Hartmann et al. 2004*b*).

2.1 Elements in XML and Constructors

The structure of XML documents is prescribed by a document type definition (DTD) (Abiteboul et al. 2000) or (almost equivalently) by an XML schema (World Wide Web Consortium (W3C) 2001). 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 2.1 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 λ 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 2.2 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 $X_1(X'_1) \oplus \cdots \oplus X_n(X'_n) \in \mathcal{N}$.

We call λ a *null attribute*, $X(X'_1, \dots, X'_n)$ a *record attribute*, $X[X']$ a *list attribute*, and $X_1(X'_1) \oplus \cdots \oplus X_n(X'_n)$ 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 will be represented by the nested attribute $X[Y]$, a sequence element definition by $X(Y_1, \dots, Y_n)$, and an alternative element definition by $X(X_1(Y_1) \oplus \cdots \oplus X_n(Y_n))$ with some new invented labels $X_1, \dots, X_n$. Furthermore, as the plus-operator in regular expressions can be expresed by the Kleene-star, an element definition 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 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 2.3 For each nested attribute $X \in \mathcal{N}$ we get a *domain* $dom(X)$ as follows:

- $dom(\lambda) = \{\top\}$;
- $dom(X(X'_1, \dots, X'_n)) = \{(X_1 : v_1, \dots, X_n : v_n) \mid v_i \in dom(X'_i)$ for $i = 1, \dots, n\}$ with labels X_i for the attributes X'_i;
- $dom(X[X']) = \{[v_1, \dots, v_n] \mid v_i \in dom(X')$ for $i = 1, \dots, n\}$, i.e. each element in $dom(X[X'])$ is a finite list with elements in $dom(X')$;
- $dom(X_1(X'_1) \oplus \cdots \oplus X_n(X'_n)) = \{(X_i : v_i) \mid v_i \in dom(X'_i)$ 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 2.2 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 2.2. 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 2.3. 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 X we define $dom(X')$ as in Definition 2.3 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 (Abiteboul, Hull & Vianu 1995, Paredaens et al. 1989) we considered the powerset $\mathcal{P}(R)$ for a relationschema 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 2.4 $\equiv$ 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 σ;
- $X_1(X'_1) \oplus \cdots \oplus X_n(X'_n) \equiv X_{\sigma(1)}(X'_{\sigma(1)}) \oplus \cdots \oplus X_{\sigma(n)}(X'_{\sigma(n)})$ for any permutation σ;
- $X(X'_1, \dots, X'_n) \equiv X(Y_1, \dots, Y_n)$ iff $X'_i \equiv Y_i$ for all $i = 1, \dots, n$;
- $X_1(X'_1) \oplus \cdots \oplus X_n(X'_n) \equiv X_1(Y_1) \oplus \cdots \oplus X_n(Y_n)$ iff $X'_i \equiv Y_i$ for all $i = 1, \dots, n$;
- $X[X'] \equiv X[Y]$ iff $X' \equiv Y$;
- $X(X'_1, \dots, Y_1(Y'_1) \oplus \cdots \oplus Y_m(Y'_m), \dots, X'_n) \equiv Y_1(X'_1, \dots, Y'_1, \dots, X'_n) \oplus \cdots \oplus Y_m(X'_1, \dots, Y'_m, \dots, X'_n)$.

Basically, the equivalence definition (apart from the last case) states that λ in record attributes can be added or removed, and that order in record and union attributes does not matter. The last case in Definition 2.4 covers an obvious restructuring rule, which was already introduced in (Abiteboul & Hull 1988).

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 2.5 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 σ and $Y_i \geq X'_i$ for all $i = 1, \dots, n$;
- $X[Y] \geq X[X']$ iff $Y \geq X'$;
- $X[X_1(X'_1) \oplus \cdots \oplus X_n(X'_n)] \geq X(X_1[X'_1], \dots, X_n[X'_n])$;
- $X(X_{i_1}[\lambda], \dots, X_{i_k}[\lambda]) \geq X_{\{i_1, \dots, i_k\}}[\lambda]$.

We should remark here that with restructuring a little bit more care is needed regarding the used labels. In fact, whenever we have a union inside a list, the labels for the union should be used as an index for the list labels, otherwise, the subattribute order $\geq$ may lead to confusion (Sali & Schewe 2004). For our purposes here we dispense with this subtlety and assume that the context in which such attributes appear helps to avoid the confusion.

Note that the last two cases in Definition 2.5 covers the restructuring for lists of unions. We have to add a little remark on notation here. As $X[X_1(X'_1) \oplus \cdots \oplus X_n(X'_n)]$ has subattributes $X(X_{i_1}[X'_{i_1}], \dots, X_{i_k}[X'_{i_k}])$, which are subattributes of $X[X_{i_1}(X'_{i_1}) \oplus \cdots \oplus X_{i_k}(X'_{i_k})]$ as well, we obtain subattributes of the form "$X[\lambda]$" for all subsets of indices. In order to distinguish these subattributes from each other, we use new labels X_I and write $X_I[\lambda]$.

Further note that due to the restructuring rules in Definitions 2.4 and 2.5 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.

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

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

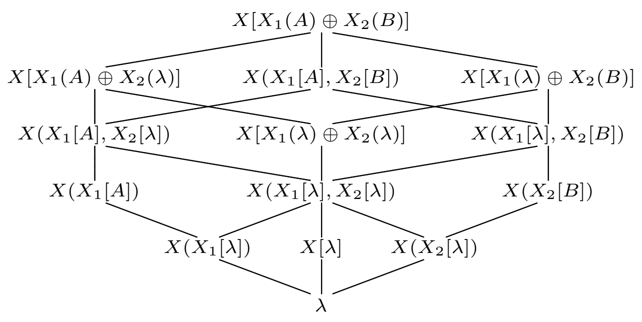
$X[X_1(A) \oplus X_2(B)]$
$X[X_1(A) \oplus X_2(\lambda)]$
$X(X_1[A], X_2[B])$
$X[X_1(\lambda) \oplus X_2(B)]$
$X(X_1[A], X_2[\lambda])$
$X[X_1(\lambda) \oplus X_2(\lambda)]$
$X(X_1[\lambda], X_2[B])$
$X(X_1[A])$
$X(X_1[\lambda], X_2[\lambda])$
$X(X_2[B])$
$X(X_1[\lambda])$
$X[\lambda]$
$X(X_2[\lambda])$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

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

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 $Y \leftarrow Z = \sqcap\{U \mid U \cup Y \geq Z\}$ exists.

Proposition 2.6 *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 2.5, and λ and X are the zero and one.*

3 Functional Dependencies in the Presence of Lists

In this section we will define functional dependencies on $\mathcal{S}(X)$ and derive some sound and complete 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 3.1 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 Σ be a set of FDs defined on some $\mathcal{S}(X')$, where X' is a nested attribute that occurs inside a given nested attribute X. A FD ψ is implied by Σ (notation: $\Sigma \models \psi$) iff all instances r with $r \models \varphi$ for all $\varphi \in \Sigma$ also satisfy ψ. As usual we write $\Sigma^* = \{\psi \mid \Sigma \models \psi\}$.

As usual we write Σ^+ for the set of all FDs that can be derived from Σ 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.

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 3.2 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 3.3 ((Hartmann et al. 2004*b*)) *The following axioms and rules are sound and complete for the implication of FDs:*

- *the λ 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 $I \cap J = \emptyset$
  $X : \{X_I[\lambda], X_J[\lambda], X_{I \cap J}[\lambda]\} \to \{X_{(I-J) \cup (J-I)}[\lambda]\}$
  and $X : \{X_I[\lambda], X_J[\lambda], X_{(I-J) \cup (J-I)}[\lambda]\} \to \{X_{I \cap J}[\lambda]\}$
- *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 $\bar{\mathcal{Z}}_i = \{X_1(\lambda) \oplus \cdots \oplus X_i(Z_i) \oplus \cdots \oplus X_n(\lambda) \mid Z_i \in \mathcal{Z}_i\}$*

The rules in Theorem 3.3 may look complicated at first glance. However, it is not wrong to say that they are almost like the Armstrong axiomatisation for FDs in the relational model. In fact, the reflexivity axiom, the extension rule and the transitivity rule reappear our rule system, whereas all other axioms and rules basically deal with structure as implied by the use of record, list and union constructors and the restructuring rules.

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

4 The Higher Level Normal Form

Let us now switch from the purely theoretical discussion of functional dependencies to the more applied questions of design quality. That is, what we want to achieve is a syntactic characterisation of semantically desirable properties. For the latter ones we concentrate on absence of redundancies. As for the relational datamodel some form of redundancy can be related to the presence of functional dependencies. The *higher-level normal form* (HLNF) (Link & Hartmann 2004) provides a characterisation for the absence of such redundancies. We will present this theory here following (Link & Hartmann 2004).

However, the theory is not yet as advanced with respect to the normal forms as it is with respect to the dependencies. That is, HLNF has only been studied so far for the case that the union constructor is excluded. Furthermore, no FDs on embedded attributes have yet been taken into account. Unfortunately, this means that it does not yet cover all the problems we would like to get covered.

If the union constructor is omitted, all subattributes will become semi-disjoint to each other. This implies that $\pi^X_Y(t_1) = \pi^X_Y(t_2)$ holds for all $Y \in \mathcal{Y}$ iff $\pi^X_{\bigsqcup \mathcal{Y}}(t_1) = \pi^X_{\bigsqcup \mathcal{Y}}(t_2)$ holds. Consequently, instead of considering FDs $\mathcal{Y} \to \mathcal{Z}$ we may restrict our attention to singleton sets of subattributes, i.e. we consider only FDs of the form $X : Y \to Z$ with $Y, Z \in \mathcal{S}(X)$.

Then it is easy to see that a FD $X : Y \to Z$ is *trivial*, i.e. it holds on all instances, iff $Y \geq Z$ holds. In order to define redundancy we need, however, a slightly more complicated class of dependencies called *nasty functional dependencies* in (Link & Hartmann 2004).

Definition 4.1 Let Σ be a set of FDs on X. The *subattribute basis* $\mathcal{B}(X)$ of X is the smallest subset of $\mathcal{S}(X)$ such that each $Y \in \mathcal{S}(X)$ can be written as a join of elements in $\mathcal{B}(X)$.

Let $\Sigma_0 \subseteq \Sigma^+$ consist of all FDs $X : Y \to Z$ such that $Y \geq Z$ holds or Z is a non-maximal base attribute of X. The *set of nasty FDs* (nFDs) on X is Σ_0^+.

It will turn out that such nasty dependencies are unavoidable. The following definition of redundancy is a straightforward generalisation of the one for the RDM (Vincent 1994).

Definition 4.2 Let Σ be a set of FDs on X. Then X is *redundant with respect to* Σ iff there is some $r \subseteq dom(X)$ satisfying Σ and distinct $t_1, t_2 \in r$ such that $\pi^X_{Y \sqcup Z}(t_1) = \pi^X_{Y \sqcup Z}(t_2)$ holds for some non-nasty FD $X : Y \to Z$.

Of course we would like to design XML databases without redundancy, and we would like to be able to characterise non-redundancy in a simple syntactic way. In fact, we have to ensure non-redundancy with respect to Σ^*, and this set of dependencies may be very large. Fortunately, redundancy with respect to Σ^* coincides with redundancy with respect to Σ, as the next theorem shows.

Theorem 4.3 ((Link & Hartmann 2004)) *Let Σ be a set of FDs on X. Then X is redundant with respect to Σ iff X is redundant with respect to Σ^*.*

With these preparations we may now state the desired normal form.

Definition 4.4 Let Σ be a set of FDs on X. Then X is in *higher-level normal form* (HLNF) with respect to Σ iff for each non-nasty FD $X : Y \to Z$ in Σ^* the subattribute Y is a key.

Our last theorem rounds up this discussion. Basically, the theorem just gives us what we desired, an easy characterisation of non-redundancy by a normal form.

Theorem 4.5 ((Link & Hartmann 2004)) *Let Σ be a set of FDs on X. Then the following statements are all equivalent:*

1. *X is non-redundant with respect to Σ.*
2. *X is in HLNF with respect to Σ.*
3. *For each non-nasty FD $X : Y \to Z$ in Σ the subattribute Y is a key.*

Thus, HLNF characterises non-redundancy, while checking HLNF does not require to know Σ^*. Furthermore, the discussion in (Link & Hartmann 2004) shows that a nested attribute in HLNF can also characterise other desirable properties such as the absence of update anomalies.

5 The Case of Sets and Multisets

While XML uses order, order in databases is usually considered to be uncommon. Therefore, it is no surprise that some work on dependencies and normal forms for XML ignores the order (Arenas & Libkin 2002, Arenas & Libkin 2004). In our context using nested attributes this corresponds to using a set or multiset constructor instead of a list constructor. Alternatively, we may consider all three constructors at the same time.

Strictly speaking, in doing so we leave the grounds of XML. Whether this is only a theoretical challenge – in fact, the cases of sets and multisets turn out to be significantly more difficult than the case of lists, in particular in combination with the union constructor – or also practically relevant shall not bother us for the moment.

However, the general theory concerning the combination of all constructors is still not completed – the work in (Hartmann et al. 2004*a*) and (Hartmann et al. 2004*d*) treats subcases. Therefore, we concentrate on a subcase, in which the "trouble making" subattributes are excluded from functional dependencies. For this we follow (Sali & Schewe 2004).

5.1 Extensions to Nested Attributes

In this section we extend our model of nested attributes and investigate the structure of the set $\mathcal{S}(X)$ of subattributes of a given nested attribute X.

Definition 5.1 Extend Definition 2.2 such that the set $\mathcal{N}$ of *nested attributes* also satisfies the following property:

- for $X \in \mathcal{L}$ and $X' \in \mathcal{N}$ we have $X\{X'\} \in \mathcal{N}$ and $X\langle X' \rangle \in \mathcal{N}$.

We call $X\{X'\}$ a *set attribute* and $X\langle X'\rangle$ a *multiset attribute*. Then it is a straightforward exercise to extend the association *dom* from Definition 2.3 to sets and multisets.

Definition 5.2 For each nested attribute $X \in \mathcal{N}$ we get a *domain* $dom(X)$, which in addition to Definition 2.3 is defined as follows:

- $dom(X\{X'\}) = \{\{v_1, \dots, v_n\} \mid v_i \in dom(X')$ for $i = 1, \dots, n\}$, i.e. each element in $dom(X\{X'\})$ is a finite set with elements in $dom(X')$;
- $dom(X\langle X'\rangle) = \{\langle v_1, \dots, v_n\rangle \mid v_i \in dom(X')$ for $i = 1, \dots, n\}$, i.e. each element in $dom(X\langle X'\rangle)$ is a finite multiset with elements in $dom(X')$.

We will again define functional dependencies on subsets $r \subseteq dom(X)$ for some nested attribute $X \in \mathcal{N}$. In order to do so, we extend the partial order $\geq$ from Definition 2.5 on equivalence classes of attributes.

Definition 5.3 $\equiv$ is the smallest *equivalence relation* on $\mathcal{N}$ satisfying the properties form Definition 2.4 plus the following properties:

- $X\{X'\} \equiv X\{Y\}$ iff $X' \equiv Y$;
- $X\langle X'\rangle \equiv X\langle Y\rangle$ iff $X' \equiv Y$;
- $X\{X_1(X'_1) \oplus \cdots \oplus X_n(X'_n)\} \equiv X(X_1\{X'_1\}, \dots, X_n\{X'_n\})$;
- $X\langle X_1(X'_1) \oplus \cdots \oplus X_n(X'_n)\rangle \equiv X(X_1\langle X'_1\rangle, \dots, X_n\langle X'_n\rangle)$.

The last two cases in Definition 5.3 cover restructuring rules extending those in (Abiteboul & Hull 1988). Obviously, if we have a set of labeled elements, we can split this set into n subsets, each of which contains just the elements with a particular label, and the union of these sets is the original set. The same holds for multisets.

As soon as the partial order $\geq$ comes in, we have to be a bit more careful with the labels. In fact, the last two cases in Definition 5.3 change labels, and as a consequence may lead to semantically wrong relations between subattributes. In order to avoid these, we should indicate the factors in a disjoint union by an index. That is, the last two restructuring rules should be formally read as

- $X_{\{1,\dots,n\}}\{X_1(X'_1) \oplus \cdots \oplus X_n(X'_n)\} \equiv X(X_1\{X'_1\}, \dots, X_n\{X'_n\})$ and
- $X_{\{1,\dots,n\}}\langle X_1(X'_1) \oplus \cdots \oplus X_n(X'_n)\rangle \equiv X(X_1\langle X'_1\rangle, \dots, X_n\langle X'_n\rangle)$.

We will, however, ignore this formal subtlety as long as there is no risk of confusion.

5.2 Extensions to Subattributes

From now on we identify $\mathcal{N}$ again with the set $\mathcal{N}/_\equiv$ of equivalence classes. In particular, we will write $=$ instead of $\equiv$. 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 5.4 *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 conditions from Definition 2.5 and following properties:*

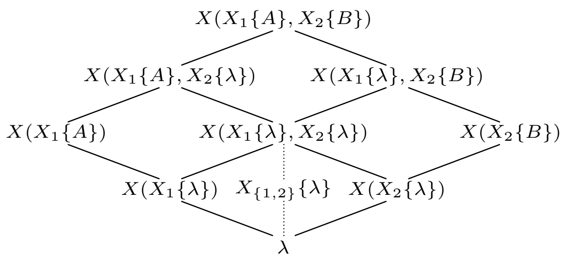

Figure 2: The lattice $\mathcal{S}(X\{X_1(A) \oplus X_2(B)\})$

- *$X\{Y\} \geq X\{X'\}$ iff $Y \geq X'$;*
- *$X\langle Y\rangle \geq X\langle X'\rangle$ iff $Y \geq X'$;*
- *$X(X_{i_1}\{\lambda\}, \dots, X_{i_k}\{\lambda\}) \geq X_{\{i_1,\dots,i_k\}}\{\lambda\}$;*
- *$X(X_{i_1}\langle\lambda\rangle, \dots, X_{i_k}\langle\lambda\rangle) \geq X_{\{i_1,\dots,i_k\}}\langle\lambda\rangle$.*

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

Note that the last two cases in Definition 5.4 cover further restructuring rules due to the union constructor. We have to add a little remark on notation here. As we identify $X\{X_1(X'_1) \oplus \cdots \oplus X_n(X'_n)\}$ with $X(X_1\{X'_1\}, \dots, X_n\{X'_n\})$, we obtain subattributes $X(X_{i_1}\{X'_{i_1}\}, \dots, X_{i_k}\{X'_{i_k}\})$ for each subset $I = \{i_1, \dots, i_k\} \subseteq \{1, \dots, n\}$. As we can also identify such a subattribute with $X\{X_{i_1}(X'_{i_1}) \oplus \cdots \oplus X_{i_k}(X'_{i_k})\}$, we obtain subattributes of the form "$X\{\lambda\}$" for all such subsets of indices. In order to distinguish these subattributes from each other, we use new labels X_I and write $X_I\{\lambda\}$. Note that $X_\emptyset\{\lambda\} = \lambda$, $X_{\{1,\dots,n\}}\{\lambda\} = X\{\lambda\}$ and $X_{\{i\}}\{\lambda\} = X(X_i\{\lambda\})$. Analogous conventions apply to multiset attributes.

Note that the presence of the restructuring rules allows us to assume that the union constructor only appears inside a set, multiset or list constructor or as the outermost constructor in a nested attribute X.

We use the notation $\mathcal{S}(X) = \{Z \in \mathcal{N} \mid X \geq Z\}$ for the *set of subattributes* of a nested attribute X. Figure 2 shows the subattributes of $X\{X_1(A) \oplus X_2(B)\}$ together with the relation $\geq$ on them.

It is a straightforward exercise to show that $\mathcal{S}(X)$ carries the structure of a lattice with zero and one and relative pseudo-complements, i.e. we have joins $Y \sqcup Z$, meets $Y \sqcap Z$ and relative pseudo-complements $Y \leftarrow Z = \sqcap\{U \mid U \sqcup Y \geq Z\}$. The zero element is λ, while the one element is X.

This means that Proposition 2.6 still holds with the extensions introduced in this section. However, $\mathcal{S}(X)$ is not a *Brouwer algebra*, as we do not have distributivity.

Therefore, in the following we consider a sublattice $\mathcal{S}^r(X) \subseteq \mathcal{S}(X)$, in which all subattributes containing some $X_I\{\lambda\}$, $X_I[\lambda]$ or $X_I\langle\lambda\rangle$ with $|I| \geq 2$ are omitted. Then it is easy to see that this restriction leads indeed to a Brouwer algebra. The dotted parts in Figure 2 indicate the parts of the lattice $\mathcal{S}(X)$ that will not be part of $\mathcal{S}^r(X)$. Furthermore, restricting our attention to $\mathcal{S}^r(X)$ implies that we may assume that the union constructor can only appear as the outermost constructor or inside a list constructor.

5.3 Counter-Free Functional Dependencies

In this section we will define counter-free functional dependencies on $\mathcal{S}(X)$ and derive some sound derivation rules. We will use sets $r \subseteq dom(X)$, which we will call simply *instances* of X.

Definition 5.5 Let $X \in \mathcal{N}$. A *counter-free functional dependency* (cfFD) on $\mathcal{S}(X)$ is an expression $X : \mathcal{Y} \to \mathcal{Z}$ with $\mathcal{Y}, \mathcal{Z} \subseteq \mathcal{S}^r(X)$.

An instance r of X *satisfies the cfFD* $X : \mathcal{Y} \to \mathcal{Z}$ (notation: $r \models \mathcal{Y} \to \mathcal{Z}$) iff for all $t_1, t_2 \in r$ with $\pi^X_Y(t_1) = \pi^X_Y(t_2)$ for all $Y \in \mathcal{Y}$ we also have $\pi^X_Z(t_1) = \pi^X_Z(t_2)$ for all $Z \in \mathcal{Z}$.

We do not have to extend the definition of semi-disjointness, so Definition 3.2 is literally what we need in the extended setting of this section. That is, for the set- and multiset-constructor we can only obtain semi-disjointness for subattributes in a $\geq$-relation. With the notion of semi-disjointness we can formulate axioms and rules for cfFDs and show their soundness and completeness.

Theorem 5.6 ((Sali & Schewe 2004)) *The following axioms and rules are sound and complete for the implication of cfFDs on $\mathcal{S}(X)$:*

- *reflexivity axiom: $X : \mathcal{Y} \to \mathcal{Z}$ for $\mathcal{Z} \subseteq \mathcal{Y}$*
- *λ axiom: $X : \emptyset \to \{\lambda\}$*
- *subattribute axiom: $X : \{Y\} \to \{Z\}$ for $Y \geq Z$*
- *join axiom: $X : \{Y, Z\} \to \{Y \sqcup Z\}$ for semi-disjoint Y, Z*
- *extension rule: $X : \mathcal{Y} \to \mathcal{Z}$ implies $\mathcal{Y} \to \mathcal{Y} \cup \mathcal{Z}$*
- *transitivity rule: $X : \mathcal{Y} \to \mathcal{Z}$ and $X : \mathcal{Z} \to \mathcal{U}$ imply $X : \mathcal{Y} \to \mathcal{U}$*

Note that we did not include lifting rules in Theorem 5.6. The reason is that the completeness proof in (Sali & Schewe 2004) does not yet hold for the case of starting with cfFDs that are defined on embedded nested attributes X'. However, it is likely that this is a straightforward extension. In this case we need the lifting rules from Theorem 3.3 plus the following two (restricted) lifting rules for sets and multisets:

- the set lifting rule: $X' : \{Y\} \to \mathcal{Z}$ implies $X\{X'\} : \{X\{Y\}\} \to \{X\{Z\} \mid Z \in \mathcal{Z}\}$
- the multiset lifting rule: $X' : \{Y\} \to \mathcal{Z}$ implies $X\langle X'\rangle : \{X\langle Y\rangle\} \to \{X\langle Z\rangle \mid Z \in \mathcal{Z}\}$

6 Keys and Armstrong Instances

In this section the idea of *keys* is generalised for the XML-like datamodel following (Sali 2004, Sali & Schewe 2004). In the relational model a *subset* $\mathcal{K}$ of attributes is a *key* iff $\mathcal{K} \to \mathcal{X}$. This could be interpreted in two ways if the relational model is considered as a special case of the higher-order model used here. That is, the *nested attribute* $X(A_1, A_2, \dots, A_n)$, where A_i's are simple attributes is equivalent with the classical relational schema $\mathcal{R}(A_1, A_2, \dots, A_n)$. A subset of attributes $\mathcal{K} = \{A_{i_1}, A_{i_2}, \dots, A_{i_r}\}$ could be identified with the *set* of subattributes $\{X(A_{i_1}), X(A_{i_2}), \dots, X(A_{i_r})\}$ or with the subattribute $X(A_{i_1}, A_{i_2}, \dots, A_{i_r})$. While the first one considers subsets of the lattice $\mathcal{S}(X)$, the second refers to elements of it. As it will be seen, these are two different manifestations of the same general concept.

Definition 6.1 Let X be a nested attribute with subattribute lattice $\mathcal{S}(X)$. A subset $\mathcal{K} \subseteq \mathcal{S}^r(X)$ is a *key* iff $\mathcal{K} \to \mathcal{S}(X)$ holds. That is, in an *instance* of $\mathcal{S}(X)$ there exists no two complex values t_1 and t_2 such that $\pi^X_K(t_1) = \pi^X_K(t_2)$ for all $K \in \mathcal{K}$.

Note that we restrict our attention here to "counter-free" keys $\mathcal{K} \subseteq \mathcal{S}^r(X)$. In (Sali 2004) a more general case has been studied, but it depends on a generalisation of a key result from (Hartmann et al. 2004*d*), which is still unproven.

Definition 6.2 An *ideal* for a nested attribute X is a subset $\mathcal{G} \subseteq \mathcal{S}^r(X)$ with $\lambda \in \mathcal{G}$ and whenever $Y \in \mathcal{G}$, $Z \in \mathcal{S}^r(X)$ with $Y \geq Z$, then also $Z \in \mathcal{G}$.

If in addition $Y \sqcup Z \in \mathcal{G}$ holds for all semi-disjoint $Y, Z \in \mathcal{G}$, we call $\mathcal{G}$ a *higher-level ideal* (or *HL-ideal*).

It is not too difficult to show that HL-ideals occur naturally as sets of attributes on which two complex values coincide.

Now let $\mathcal{K}$ be key and consider the HL-ideal $hl(\mathcal{K})$ generated by $\mathcal{K}$. Clearly, $hl(\mathcal{K})$ is also a key. This converse is also true.

Theorem 6.3 ((Sali 2004)) *Let $\mathcal{H}$ be an HL-ideal of $\mathcal{S}(X)$, which is a key. Let $\mathcal{G} \subseteq \mathcal{H}$ be any generating subset of $\mathcal{H}$. Then $\mathcal{G} \to \mathcal{S}(X)$ also holds, i.e. $\mathcal{G}$ is also a key.*

Theorem 6.3 allows us to identify keys with their generated HL-ideals, and systems of keys with families of HL-ideals. Two keys are said to be *equivalent*, if they generate the *same* HL-ideal. We introduce an ordering on the (equivalence classes of) keys as follows

$$\mathcal{K}_1 \preceq \mathcal{K}_2 \iff hl(\mathcal{K}_1) \subseteq hl(\mathcal{K}_2). \qquad (1)$$

A key is said to be *minimal*, if it is minimal under the ordering $\preceq$. Thus, a system $\mathfrak{K}$ of minimal keys corresponds to a *Sperner* family $\mathfrak{H}$ of HL-ideals.

Example 6.1 Let $X = X(A_1, A_2, \dots, A_n)$ be a nested attribute with simple attributes A_i. Then *any* pair of subattributes are semi-disjoint, so *any* HL-ideal is a principal ideal. The HL-ideal generated by $\{X(A_{i_1}), X(A_{i_2}), \dots, X(A_{i_r})\}$ happens to be the principal ideal generated by $X(A_{i_1}, A_{i_2}, \dots, A_{i_r})$. This shows that Definition 6.1 is a sound generalisation of the classical relational case.

Let $\mathcal{K}$ be a key and $\mathcal{H} = hl(\mathcal{K})$. The maximal elements of $\mathcal{H}$ under the ordering $\leq$ of $\mathcal{S}(X)$ form a generating set of $\mathcal{H}$, which is called the *canonical* generating set of $\mathcal{H}$. Whenever it is not stated explicitly otherwise, we will consider the canonical generating set of an HL-ideal.

Definition 6.4 A set $\mathcal{A} \subset \mathcal{S}^r(X)$ of subattributes is called an *antikey* iff $\mathcal{A} \not\twoheadrightarrow \mathcal{S}(X)$.

Using Theorem 6.3 we can prove the equivalence of antikeys and HL-ideals.

Theorem 6.5 ((Sali 2004)) *Let $\mathcal{A}$ be an antikey and let $\mathcal{G} = hl(\mathcal{A})$ be the HL-ideal generated by $\mathcal{A}$. Then $\mathcal{G}$ (and consequently any generating set of it) is an antikey.*

The ordering $\preceq$ defined by (1) can be extended to (equivalence classes of) antikeys, as well. The *maximum* antikeys form a Sperner family $\mathfrak{A}$ of HL-ideals. Let $\mathfrak{K} = \{\mathcal{K}_1, \mathcal{K}_2, \dots, \mathcal{K}_r\}$ be the system of minimal keys. Then the set of maximum antikeys is

$$\mathfrak{A} = \{\mathcal{A}_1, \mathcal{A}_2, \dots, \mathcal{A}_s\} \qquad (2)$$

such that $\mathcal{K}_i \not\subseteq \mathcal{A}_j$ for all pairs i, j and the $\mathcal{A}_j$ are maximal under this condition.

Let $\mathfrak{K} = \{\mathcal{K}_1, \mathcal{K}_2, \dots, \mathcal{K}_r\}$ be a Sperner system of HL-ideals, furthermore let $\mathfrak{A} = \{\mathcal{A}_1, \mathcal{A}_2, \dots, \mathcal{A}_s\}$ be the Sperner system of HL-ideals defined by (2). In this case $\mathfrak{A}$ is denoted by $\mathfrak{A} = \mathfrak{K}^{-1}$.

The question is *which Sperner systems of HL-ideals occur as families of minimal keys*? In the relational model the answer was given by Armstrong and Demetrovics (Demetrovics 1979). Namely they proved that *every* Sperner-family of subsets of attributes can be a system of minimal keys. However, the analogous statement does not hold in general for the higher order datamodel. An example was given in (Sali 2004), however that works in $\mathbb{S}(X)$ but not in $\mathbb{S}^r(X)$. The following example works in the counter-free case, as well.

Example 6.2 Consider the nested attribute $X\{X_1(A_1) \oplus X_2(A_2) \oplus \ldots \oplus X_n(A_n)\}$ $(n > 5)$ and the Brouwer-algebra $\mathbb{S}^r(X)$. Let $\mathfrak{K} = \{\mathcal{K}_{ij} : 1 \le i < j \le n\}$ be the Sperner-family of HL-ideals, where $\mathcal{K}_{ij}$ is the principal ideal generated by $X(X_i\{\lambda\}, X_j\{\lambda\})$ for $1 \le i < j \le n$. It is not hard to see that the system $\mathfrak{A}$ of maximal antikeys belonging to $\mathfrak{K}$ is $\mathfrak{A} = \mathfrak{K}^{-1} = \{\mathcal{A}_i : 1 \le i \le n\}$ where $\mathcal{A}_i$ is the principal ideal generated by $X(X_i\{A_i\})$.

Suppose, that the system $\mathfrak{K}$ has an Armstrong instance r as defined in Definition 6.6. This means, that no two tuples of r can agree on $X(X_i\{\lambda\}, X_j\{\lambda\})$, for any $1 \le i < j \le n$, but there are tuples t_0^i, t_1^i for all $1 \le i \le n$ such that $\pi^X_{X(X_i\{A_i\})}(t_0^i) = \pi^X_{X(X_i\{A_i\})}(t_1^i)$. Since $\pi^X_{X(X_i\{\lambda\},X_j\{\lambda\})}(t_0^i) \neq \pi^X_{X(X_i\{\lambda\},X_j\{\lambda\})}(t_1^i)$, we obtain that for any $j \neq i$ exactly one of t_0^i and t_1^i has a nonempty component of type $(X_j : a_j)$. Consider $t_0^1, t_1^1, t_0^2, t_1^2$, and let us define two sets of indices as $I = \{j : t_0^1 \text{ has component } (X_j : a_j)\}$,and $J = \{j : t_0^2 \text{ has component } (X_j : a_j)\}$. We may assume without loss of generality that $|J| > 2$. It follows that either $|I \cap J| \ge 2$ or $|(\{2, 3, \ldots, n\} \setminus I) \cap J| \ge 2$. In the first case t_0^1 and t_0^2 agree on some $X(X_i\{\lambda\}, X_j\{\lambda\})$ (both being $(\top, \top)$), while in the second case t_0^1 and t_1^2 do, (both being $(\emptyset, \emptyset)$). This contradicts to the assumption that $\mathfrak{K} = \{\mathcal{K}_{ij} : 1 \le i < j \le n\}$ is a system of minimal keys.

Definition 6.6 Let $\mathfrak{K} = \{\mathcal{K}_1, \mathcal{K}_2, \ldots, \mathcal{K}_r\}$ be a Sperner system of HL-ideals. r is called an *Armstrong instance* for $\mathfrak{K}$ if the minimal key system determined by r is $\mathfrak{K}$. If there exists an Armstrong instance for a given $\mathfrak{K}$, then $s(\mathfrak{K})$ denotes the smallest possible number of complex values in an Armstrong instance of $\mathfrak{K}$. Otherwise define $s(\mathfrak{K}) = \infty$.

Consider $X = X_1(X_1') \oplus X_2(X_2') \oplus \ldots \oplus X_n(X_n')$. An instance r of X can be partitioned $r = r_1 \dot\cup r_2 \dot\cup \ldots \dot\cup r_n$ where r_i consists of the complex values of type $(X_i : v_i)$. A set $\mathcal{K} = \{Y_1, Y_2, \ldots, Y_m\} \subseteq \mathcal{S}(X)$ is a key iff $\pi^X_{X_i(X_i')}(\mathcal{K})$ is a key in $\mathcal{S}(X_i)$ for all $1 \le i \le n$. Note that if $Y_j = X_1(Y_1^j) \oplus \ldots \oplus X_n(Y_n^j)$, then $\pi^X_{X_i(X_i')}(\mathcal{K}) = \{X_i(Y_i^1), X_i(Y_i^2), \ldots, X_i(Y_i^m)\}$. This observation allows us to consider only those cases where the union constructor occurs only inside a set, list or multiset constructor.

Theorem 6.7 ((Sali & Schewe 2004)) *Let X be a nested attribute and assume that the union constructor occurs in X only inside of a set, multiset or list constructor. Let $\mathfrak{K} = \{\mathcal{K}_1, \mathcal{K}_2, \ldots \mathcal{K}_n\}$ be a Sperner family of HL-ideals of $\mathcal{S}(X)$. Let us assume that for all $1 \le i \le n$, the canonical generating set of $\mathcal{K}_i$ contains a subattribute that has a simple attribute in its construction or a list or multiset subattribute. Then $\mathfrak{K}$ has an Armstrong instance and $s(\mathfrak{K}) \le 2|\mathfrak{K}^{-1}|$.*

The condition in Theorem 6.7 is sufficient, but not necessary for the existence of Armstrong instance. Consider the following example.

Example 6.3 Take $\mathcal{S}(X\{X_1(A) \oplus X_2(B)\})$ shown on Figure 2.

Let $\mathfrak{K} = \{hl(\{X(X_1\{\lambda\}, X_2\{\lambda\})\})\}$, then $\mathfrak{K}^{-1}$ consists of two HL-ideals, $\mathcal{A}_1 = hl(X(X_1\{A\}))$ and $\mathcal{A}_2 = hl(X(X_2\{B\}))$. Then three complex values are constructed, namely $t_0^1 = t_0^2 = \emptyset$, $t_1^1 = \{(X_2 : b)\}$, and $t_1^2 = \{(X_1 : a)\}$. These three complex values clearly form an Armstrong instance for $\mathfrak{K}$.

Further cases and examples have been studied in (Sali & Schewe 2004).

7 Where from Here? – An Agenda for Further Research

In this paper we gave a brief overview of some the research developments regarding dependency and normal form theory for XML databases. We concentrated on the approach that build (rational) trees from constructors for records, lists, sets, multisets and unions. Furthermore, we only looked at functional dependencies and keys.

From the presentation it is clear that even for FDs there are still a lot of open research problems, e.g.

- What is the exact relationship between the various approaches by Arenas & Libkin, Vincent et al., and Hartmann, Link, Sali & Schewe? In particular, why do some definitions of FDs lead to axiomatisability, while others do not? Is there a unifying framework that captures all approaches?
- How does the axiomatisation look like, if the full set of constructors is taken into account and all FDs, not just a restricted class, are considered? According to (Hartmann et al. 2004*d*) there seems to be a positive answer to axiomatisability for weak FDs, but how can this be generalised to FDs knowing that the classical proof does no longer work in this case?
- How are the normal forms in (Link & Hartmann 2004, Arenas & Libkin 2004, Vincent et al. 2004) related to each other? Is there a generalisation that captures all of them and in particular captures all constructors and rational trees?
- What are efficient algorithms for detecting a normal form violation and achieving normal forms?
- Are there other desirable properties that might be destroyed by normalisation? If yes, what are weaker normal forms that make a compromise between these properties.

For keys, the work in (Sali 2004, Sali & Schewe 2004) is also only a starting point. In fact, it does not show more than a significant disparity with the results for the relational model. For instances, not all Sperner systems of HL-ideals define keys, not even the maximal one. Thus, much more sophisticated investigations are needed here as well.

Finally, there are of course many other classes of dependencies (Thalheim 1991) that require a generalisation to XML-like databases. For multi-valued dependencies (MVDs) some work has been started in (Hartmann, Link & Schewe 2004*c*, Vincent & Liu 2003*b*, Vincent & Liu 2003*c*), but has not yet reached the same number of result as the work on FDs. The same open questions as for FDs can also be asked for MVDs.

In fact, all these problems are problems of database theory that already came up with complex value databases and object-oriented databases. However, the almost dying interest in these areas has blocked fruitful research on these problems. With XML, however, there might be hope of a long-lasting practical interest, which will enable tackling all these theoretical research challenges.

References

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

Abiteboul, S. & Hull, R. (1988), 'Restructuring hierarchical database objects', *Theoretical Computer Science* **62**(1-2), 3–38.

Abiteboul, S., Hull, R. & Vianu, V. (1995), *Foundations of Databases*, Addison-Wesley.

Abiteboul, S., Quass, D., McHugh, J., Widom, J. & Wiener, J. (1997), 'The LOREL query language for semi-structured data', *International Journal on Digital Libraries* **1**(1), 68–88.

Arenas, M. & Libkin, L. (2002), A normal form for XML documents, *in* 'PODS 2002', ACM.

Arenas, M. & Libkin, L. (2004), 'A normal form for XML documents', *ACM ToDS* **29**(1), 195–232.

Bohannon, P., Freire, J., Roy, P. & Simeon, J. (2002), From XML-schema to relations: A cost-based approach to XML storage, *in* 'International Conference on Data Engineering', San José, California, pp. 64–75.

Buneman, P., Davidson, S., Fan, W., Hara, C. & Tan, W. (2001), Keys for XML, *in* 'Tenth WWW Conference', IEEE.

Buneman, P., Davidson, S., Fernandez, M. & Suciu, D. (1997), Adding structure to unstructured data, *in* 'International Conference on Database Theory – ICDT'97', Vol. 1186 of *Lecture Notes in Computer Science*, Springer-Verlag, Delphi, Greece, pp. 336–350.

Buneman, P., Davidson, S., Hillebrand, G. & Suciu, D. (1996), A query language and optimization techniques for unstructured data, *in* 'Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data', Montréal, Canada, pp. 505–516.

Cluet, S., Delobel, C., Siméon, J. & Smaga, K. (1998), Your mediators need data conversion!, *in* 'Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data', Seattle, Washington, pp. 177–188.

Demetrovics, J. (1979), 'On the equivalence of candidate keys with Sperner systems', *Acta Cybernetica* **4**, 247–252.

Deutsch, A., Fernandez, M., Florescu, D., Levy, A. & Suciu, D. (1999), 'A query language for XML', *Computer Networks* **31**(11-16), 1155–1169.

Deutsch, A., Fernandez, M. & Suciu, D. (1999), Storing semistructured data with STORED, *in* 'Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data', Philadelphia, Pennsylvania, pp. 431–442.

Fan, W. & Libkin, L. (2001), On XML integrity constraints in the presence of DTDs, *in* 'PODS 2001', ACM.

Florescu, D. & Kossmann, D. (1999), 'Storing and querying XML data using an RDBMS', *Bulletin of the Technical Committee on Data Engineering* pp. 27–34.

Goldfarb, C. F. & Prescod, P. (1998), *The XML Handbook*, Prentice Hall, New Jersey.

Hartmann, S. & Link, S. (2003), 'On functional dependencies in advanced data models', *Electronic Notes in Theoretical Computer Science* **84**.

Hartmann, S., Link, S. & Schewe, K.-D. (2004*a*), Axiomatisation of functional dependencies in the presence of records, lists, sets and multisets, Technical Report 1/2004, Massey University, Department of Information Systems.

Hartmann, S., Link, S. & Schewe, K.-D. (2004*b*), Functional dependencies over XML documents with DTDs. submitted for publication.

Hartmann, S., Link, S. & Schewe, K.-D. (2004*c*), Reasoning about functional and multi-valued dependencies in the presence of lists, *in* D. Seipel & J. M. Turull Torres, eds, 'Foundations of Information and Knowledge Systems', Vol. 2942 of *LNCS*, Springer Verlag, pp. 134–154.

Hartmann, S., Link, S. & Schewe, K.-D. (2004*d*), Weak functional dependencies in higher-order datamodels, *in* D. Seipel & J. M. Turull Torres, eds, 'Foundations of Information and Knowledge Systems', Vol. 2942 of *LNCS*, Springer Verlag, pp. 116–133.

Link, S. & Hartmann, S. (2004), Normalisation in the presence of lists, *in* K.-D. Schewe & H. Wiliams, eds, 'Database Technologies 2004 – Proceedings ADC 2004', Vol. 27 of *CRPIT*, Australian Computer Society, pp. 49–60.

Mok, W. Y., Ng, Y. K. & Embley, D. W. (1996), 'A normal form for precisely charachterizing redundancy in nested relations', *Transactions on Database Systems* **21**, 77–106.

Özsoyoglû, Z. M. & Yuan, L. Y. (1987), 'A new normal form for nested relations', *Transactions on Database Systems* **12**, 111–136.

Paredaens, J., De Bra, P., Gyssens, M. & Van Gucht, D. (1989), *The Structure of the Relational Database Model*, Springer-Verlag.

Sali, A. (2004), Minimal keys in higher-order datamodels, *in* D. Seipel & J. M. Turull Torres, eds, 'Foundations of Information and Knowledge Systems', Vol. 2942 of *LNCS*, Springer Verlag, pp. 242–251.

Sali, A. & Schewe, K.-D. (2004), Counter-free keys and functional dependencies in higher-order datamodels, Technical Report 7/2004, Massey University, Department of Information Systems.

Schmidt, A., Kersten, M., Windhouwer, M. & Waas, F. (2000), Efficient relational storage and retrieval of XML documents, *in* 'The World-Wide Web and Databases – Third International Workshop', Vol. 1997 of *Lecture Notes in Computer Science*, Springer-Verlag, Dallas, Texas, pp. 137–150.

Shanmugasundaram, J., Tufte, K., Zhang, C., He, G., De Witt, D. J. & Naughton, J. F. (1999), Relational databases for querying XML documents: Limitations and opportunities, *in* 'Proceedings of 25th International Conference on Very Large Data Bases', Edinburgh, Scotland, pp. 302–314.

Tatarinov, I., Ives, Z., Halevy, A. & Weld, D. (2001), Updating XML, *in* 'Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data', Santa Barbara, California, pp. 413–424.

Thalheim, B. (1991), *Dependencies in Relational Databases*, Teubner-Verlag.

Vincent, M. (1994), The semantic justification for normal forms in relational database design, PhD thesis, Monash University, Melbourne, Australia.

Vincent, M., Liu, J. & Liu, C. (2004), 'Dtrong functional dependencies and their application to normal forms in XML', *ACM ToDS* **29**(3), 445–462.

Vincent, M. W. & Liu, J. (2003*a*), Functional dependencies for XML, *in* 'Web Technologies and Applications: 5th Asia-Pacific Web Conference', Vol. 2642 of *LNCS*, Springer-Verlag, pp. 22–34.

Vincent, M. W. & Liu, J. (2003*b*), Multivalued dependencies and a 4NF for XML, *in* 'Advanced Information Systems Engineering: 15th International Conference CAiSE 2003', Vol. 2681 of *LNCS*, Springer-Verlag, pp. 14–29.

Vincent, M. W. & Liu, J. (2003*c*), Multivalued dependencies in XML, *in* 'British National Conference on Database Systems: BNCOD 2003', Vol. 2712 of *LNCS*, Springer-Verlag, pp. 4–18.

World Wide Web Consortium (W3C) (2001), 'XML Schema', `http://www.w3c.org/TR/xmlschema-0`, `http://www.w3c.org/TR/xmlschema-1`, `http://www.w3c.org/TR/xmlschema-2`.

World Wide Web Consortium (W3C) (2002), 'XQuery', `http://www.w3c.org/TR/xquery`.