Axiomatising Functional Dependencies for XML with Frequencies

Sven Hartmann and Thu Trinh*

Information Science Research Centre,
Massey University, Palmerston North, New Zealand
{s.hartmann, t.trinh}@massey.ac.nz

Abstract. We provide a finite axiomatisation for a class of functional dependencies for XML data that are defined in the context of a simple XML tree model reflecting the permitted parent-child relationships together with their frequencies.

1 Introduction

The question of how to represent and efficiently manage complex application data is one of the major challenges database research faces today. XML (the Extensible Markup Language) has gained popularity as a standard for exchanging data on the web. The flexibility of XML and its wide acceptance as a standard make it also a good choice for modelling heterogenous and highly structured data from various application domains. As a consequence, XML databases (in the form of data-centric XML documents) have attracted a great deal of interest.

As for the relational data model (RDM), integrity constraints are needed to capture more of the semantics of the data stored in an XML database. Several types of integrity constraints have been studied in the context of XML, with a focus on various key constraints, functional dependencies, inclusion constraints, and path constraints. For relational databases, functional dependencies have been vital in the investigation of how to design "good" database schemas to avoid or minimise problems relating to data redundancy and data inconsistency. The same problems can be shown to exist in poorly designed XML databases. Not surprisingly, functional dependencies for XML (often referred to as XFDs) have recently gained much attention.

An important problem involving XFDs is that of logical implication, i.e., deciding whether a new XFD holds, given a set of existing XFDs. This is important for minimising the cost of checking that a database satisfies a set of XFDs, and may also be helpful when XFDs are propagated to view definitions. One approach to solve this problem is to develop a sound and complete set of inference rules for generating symbolic proofs of logical implication.

For the RDM, the implication problem for functional dependencies is decidable in linear time, and the Armstrong system of inference rules is sound and

* Thu Trinh's research was supported by a Lovell and Berys Clark and a William Georgetti masterate scholarship.

J. Dix and S.J. Hegner (Eds.): FoIKS 2006, LNCS 3861, pp. 159–178, 2006.

complete. For XML the story is more complicated. Before studying XFDs and actually using them in database design, they have to be formally defined. In the literature [3, 4, 9-11, 13, 18-20, 17, 21, 22], several generalisations of functional dependencies to XML have been proposed, and they do not always reflect the same kind of dependencies in an XML database. The difficulty with XML data is that its nested structure is more complex than the rigid structure of relational data, and thus may well observe a larger variety of data dependencies.

In this paper we use an approach that considers XFDs in the context of a simple tree model: XFDs are defined over a schema tree that reflects the permitted parent-child relationships, and apply to the almost-copies of the schema tree that can be found in an XML data tree under inspection. XML schema trees capture information on the frequency of parent-child relationships, that is, they show whether a child is optional or required, and whether it is unique or may occur multiple times. This approach to XFDs has been suggested in [10], and comes close to the approach taken in [3, 4] where XFDs are defined on the basis of paths evolving from the root element in an XML document. This idea goes back to earlier studies of functional dependencies in semantic and object-oriented data models [15, 23]. It should be noted that XFDs may well interact in a non-trivial way with chosen specifications (like DTDs) as demonstrated, e.g., in [2, 4].

This paper is organised as follows. In Section 2, we provide preliminary notions like XML schema trees and data trees. In Section 3, we present an example that illustrates our approach, while we formally define XFDs in Section 4. In Section 5, we assemble sound inference rules, which are then shown to be complete in Section 6. In Section 7, we extend our investigation to account for ID-attributes that are widely used in XML databases. Finally, Section 8 gives an overview of related work and discusses similarities and differences.

2 Preliminary Notations

We start with reviewing basic features of a simple XML tree model. Within this paper, all graphs considered are directed, without parallel arcs and finite unless stated otherwise. For every graph G, let V_G denote its set of vertices and A_G its set of arcs. A *rooted graph* is a graph G with one distinguished vertex r_G, called the *root* of G, such that there is a directed path from r_G to every other vertex in V_G. A *rooted tree* is a rooted graph T without any (non-directed) cycles. A graph G is *empty* if A_G is empty. Specifically, G is an *empty rooted graph* if it consists of a single vertex r_G. For every vertex v, let $Succ_G(v)$ denote its (possibly empty) set of successors, called *children*, in G. A non-isolated vertex without children is a *leaf* of G. Let L_G denote the set of all leaves of G.

Definition 1. *Given a vertex $v \in V_G$ and a subset $W \subseteq L_G$ of leaves, a* v-subgraph *of G is the graph union of all directed walks from v to some $w \in W$. A* v-walk *of G is a directed walk from v to a single leaf w of G. Every v-walk or v-subgraph of a rooted tree is again a rooted tree.*

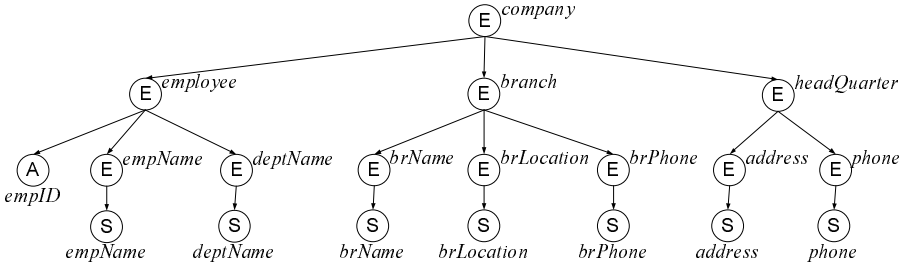


Fig. 1. An XML tree showing the names and kinds of vertices

Let $ENames$ and $ANames$ be fixed sets of element names and attribute names, respectively. Also let the symbols E, A and S reflect whether a vertex represents an element, attribute or text data respectively.

Definition 2. *An* XML graph *is a rooted graph G together with the mappings $name : V_G \to ENames \cup ANames$ and $kind : V_G \to \{E, A, S\}$ assigning every vertex its name and kind, respectively. If G is a rooted tree, then we speak of an* XML tree.

In this paper, all XML trees are assumed to be unordered trees. Let V_G^E, V_G^A and V_G^S consist of all vertices in V_G of kind E, A and S, respectively. We suppose that in an XML graph, vertices of kind A and S are always leaves and conversely all leaves are either of kind A or S, that is, $L_G = V_G^A \cup V_G^S$. Thus, within this paper, we do not consider empty elements unless G is empty.

Definition 3. *Let G' and G be two XML graphs, and consider a mapping $\phi : V_{G'} \to V_G$. ϕ is said to be* kind-preserving *if the image of a vertex is of the same kind as the vertex itself, that is, $kind(v') = kind(\phi(v'))$ for all $v' \in V_{G'}$. Further, ϕ is* name-preserving *if the image of a vertex carries the same name as the vertex itself, that is, $name(v') = name(\phi(v'))$ for all $v' \in V_{G'}$. The mapping ϕ is a* homomorphism *between G' and G if all of the following conditions hold:*

1. *the root of G' is mapped to the root of G, that is, $\phi(r_{G'}) = r_G$*
2. *every arc of G' is mapped to an arc of G, that is, $(u', v') \in A_{G'}$ implies $(\phi(u'), \phi(v')) \in A_G$*
3. *ϕ is kind-preserving and name-preserving.*

Definition 4. *A homomorphism $\phi : V_{G'} \to V_G$ is an* isomorphism *if ϕ is bijective and ϕ^{-1} is a homomorphism. Whenever such an isomorphism exists, G' is said to be* isomorphic *to G, denoted by $G' \cong G$. We also call G' a* copy *of G.*

Definition 5. *A subgraph H' of G' is a copy of a subgraph H of G if the restriction of $\phi : V_{G'} \to V_G$ to H' and H is an isomorphism between H' and H. An $r_{G'}$-subgraph H' of G' is a* subcopy of G *if it is a copy of some r_G-subgraph H of G. A maximal subcopy of G is a subcopy of G which is not an r_G-subgraph of any other subcopy of G. A maximal subcopy of G is called an* almost-copy of G.

It should be noted that all copies of G in G' are almost-copies of G, but not vice versa. Also we can observe that a homomorphism $\phi : V_{G'} \to V_G$ is not an isomorphism whenever G' contains more than one copy of G or no copy (but possibly many almost-copies) of G.

Definition 6. *An* XML schema graph *is an XML graph G together with a mapping $freq : A_G \to \{?, 1, +, *\}$ assigning every arc its frequency. Every arc $a = (v, w)$ where w is of kind A has frequency $freq(a) =?$ or 1. Every arc $a = (v, w)$ where $kind(v) = E$ and $kind(w) = S$ has frequency $freq(a) = 1$. Further, we assume no vertex in V_G has two successors with the same name and the same kind. If G is more specifically an XML tree, then we speak of an* XML schema tree.

We use "f*-arc*" to refer to an arc of frequency f and "f/g*-arc*" to refer to an arc of frequency f or g. For example, a ?-arc refers to an arc of frequency ?, while a */+-arc refers to an arc of frequency * or +. For an XML schema graph G, let $G_{\leq 1}$ be the graph union of all ?/1-arcs in A_G, and $G_{\geq 1}$ be the graph union of all 1/+-arcs in A_G. Note that $G_{\leq 1}$ and $G_{\geq 1}$ may not be r_G-subgraphs of G.

Definition 7. *An* XML data tree *is an XML tree T' together with an evaluation $val : L_{T'} \to STRING$ assigning every leaf v a (possibly empty) string $val(v)$.*

Definition 8. *Let G be an XML schema graph. An XML data tree T' is* compatible *with G, denoted by $T' \rhd G$, if there is a homomorphism $\phi : V_{T'} \to V_G$ between T' and G such that for each vertex v' of T' and each arc $a = (\phi(v'), w)$ of G, the number of arcs $a' = (v', w'_i)$ mapped to a is at most 1 if $freq(a) =?$, exactly 1 if $freq(a) = 1$, at least 1 if $freq(a) = +$, and arbitrarily many if $freq(a) = *$. Due to the definition of a schema graph, this homomorphism is unique if it exists.*

An XML schema graph may be developed by a database designer similar to a (rather simple) database schema, or it can be derived from other specifications (such as DTDs or XSDs). Alternatively, an XML schema graph may also be derived from an XML document itself, cf. [10]. At this point a short remark is called for. Given an XML data tree T', there is usually more than just a single XML schema graph G such that T' is compatible with G. For example, G may well be extended by adjoining new vertices and arcs, or by (partially) unfolding it. Recall that, in our definition of an XML schema graph, we did neither claim the vertices of kind E to have mutually distinct names, nor those of kind A. It is well-known that every rooted graph G may be uniquely transformed into a rooted tree T_G by completely unfolding it, cf. [8].

Let T'_1 be any almost-copy of T in an XML data tree $T' \rhd T$. It is possible that T'_1 does not contain a copy of some r_T-walk which contains an ?-arc or *-arc. Note that this flexibility is one of the desirable features of XML to adequately represent heterogenous data. We say that T'_1 is *missing* a copy of an r_T-walk C of T if T'_1 does not contain a copy of C, otherwise T'_1 is said to be *not missing* a copy of C. Similarly the data tree T' is said to be *missing* a copy of C if it does not contain a copy of C, and *not missing* a copy of C otherwise.

For two r_G-subgraphs X and Y of some graph G, we may use $X \subseteq Y$ to denote that X is an r_G-subgraph of Y in G, and more specifically $X \in Y$ to denote that X is an r_G-walk of Y in G.

Next we briefly discuss operators to construct new trees from given ones. Let G be an XML graph, and X, Y be subgraphs of G. The *union* of X and Y, denoted by $X \cup Y$, is the restriction of the graph union of X and Y to its maximal r_G-subgraph of G. For convenience, we sometimes omit the union symbol and write XY instead of $X \cup Y$. The *intersection* of X and Y, denoted by $X \cap Y$, is the union of all r_G-walks that belong to both X and Y. The *difference* between X and Y, denoted by $X - Y$, is the union of all r_G-walks belonging to X but not to Y. In particular, $X \cap Y$ and $X - Y$ are r_G-subgraphs of G.

The intersection operator is associative but the union and difference operators are not associative. The union and intersection operators are commutative but the difference operator is not. In the absence of parentheses, we suppose that the union and intersection operators bind tighter than the difference operator. For example, by $X \cup Y - Z$ we mean $(X \cup Y) - Z$.

Let G' and G be two XML graphs, and $\phi : V_{G'} \to V_G$ be a homomorphism between them. Given an r_G-subgraph H in G, the *projection* of G' to the subgraph H in G, denoted by $G'|_H$, is the union of all the subcopies of H in G'. The projection $G'|_H$ is an $r_{G'}$-subgraph of G'.

3 A Motivating Example

Our example describes information stored about a product development company and its employees. The company has one head quarter office contactable by postal mail or phone. Furthermore, the company operates various departments and has multiple branches.

In our example here, we use attributes rather than text elements purely to end up with more compact XML graphs. Of course, each attribute in an XML graph G with name n may alternatively be modelled by a vertex v of kind E with name n and a child $w \in Succ_G(v)$ of kind S and name n. Here, we are not concerned with the question of whether some information are better modelled as an attribute or text element.

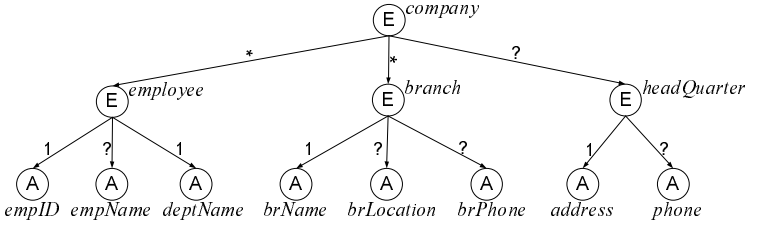


Fig. 2. The XML schema tree PD

For the ease of presentation, we have chosen examples where the leaf names are unique. In this paper, we may therefore refer to an r_G-walk to some leaf carrying the name "B" simply as $[[B]]$, e.g., $[[brPhone]]$. Further, we may refer to an r_G-subgraph X by listing the names of all leaves in X separated by white spaces, e.g., $[[brName\ brPhone]]$.

4 Functional Dependencies for XML

Two isomorphic XML data trees T' and T are said to be *value-equal*, denoted by $T' = T$, if the isomorphism $\phi : V_{T'} \to V_T$ between T' and T is evaluation-preserving, that is, $val(\phi(v')) = val(v')$ holds for every $v' \in L_{T'}$. We are now ready to present our definition of functional dependencies for XML.

Definition 9. *Given an XML schema graph T, a* functional dependency *(or XFD for short) on T is an expression $X \to Y$ where X and Y are non-empty r_T-subgraphs in T. Let T' be an XML data tree which is compatible with T and let $\phi : V_{T'} \to V_T$ be the unique homomorphism between T' and T. Then T'* satisfies *the XFD $X \to Y$, written as $\models_{T'} X \to Y$, if and only if for any two almost-copies T'_1 and T'_2 of T in T' the projections $T'_1|_Y$ and $T'_2|_Y$ are value-equal whenever the projections $T'_1|_X$ and $T'_2|_X$ are value-equal and copies of X, i.e., $T'_1|_Y = T'_2|_Y$ whenever $T'_1|_X = T'_2|_X \cong X$.*

Example 10. Suppose each department of our product development company is located at a single branch. Branches have unique phone numbers and are located in unique locations. Employees are assigned unique employee IDs. We use the following XFDs to model the PD company information:

$$
\begin{array}{lll}
(PD\_XFD1) & [[brName]] & \to [[brLocation]] \\
(PD\_XFD2) & [[brName]] & \to [[brPhone]] \\
(PD\_XFD3) & [[brLocation]] & \to [[brName]] \\
(PD\_XFD4) & [[deptName]] & \to [[brName]] \\
(PD\_XFD5) & [[empID]] & \to [[empName]]
\end{array}
$$

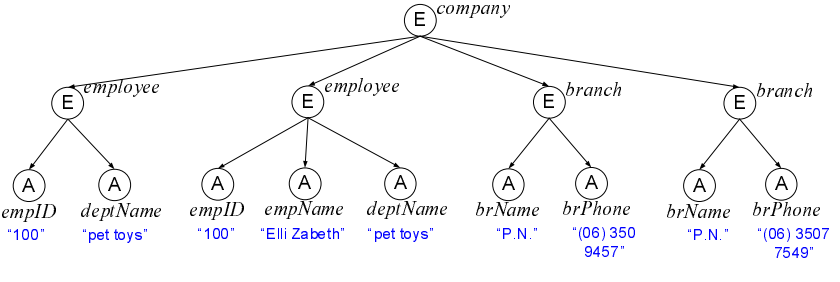


Fig. 3. An XML data tree PD′ compatible with PD

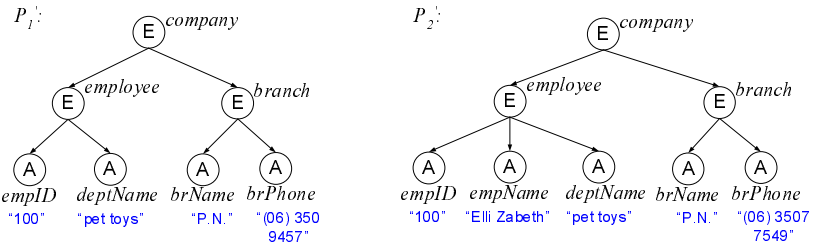


Fig. 4. Two of the four almost-copies of PD contained in PD′

The XML data tree PD′ in Figure 3 contains four almost-copies of PD, two of which are shown in Figure 4. The two remaining almost-copies can be obtained as follows:

$$P_3' = P_1'|_{[[\,empID\ deptName\,]]} \cup P_2'|_{[[\,brName\ brPhone\,]]}$$
$$P_4' = P_2'|_{[[\,empID\ empName\ deptName\,]]} \cup P_1'|_{[[\,brName\ brPhone\,]]}$$

The almost-copies P_1' and P_2' contain value-equal copies of $[[\,brName\,]]$, but they differ in their copies of $[[\,brPhone\,]]$. Therefore PD′ does not satisfy PD_XFD2. Moreover, PD′ does not satisfy PD_XFD5 since P_1' and P_2' contain value-equal copies of $[[\,empID\,]]$, but only P_2' is not missing a copy of $[[\,empName\,]]$.

On the other hand, PD_XFD3 is trivially satisfied because PD′ is missing a copy of $[[\,brLocation\,]]$. P_1' and P_2' contain value-equal copies of $[[\,brName\,]]$ and are both missing a copy of $[[\,brLocation\,]]$. Since P_3', P_4' are constructed from P_1' and P_2', it is easy to see that any two almost-copies of PD in PD′ have a value-equal copy of $[[\,brName\,]]$ and are missing a copy of $[[\,brLocation\,]]$. Hence PD′ satisfies PD_XFD1. Both P_1' and P_2' contain value-equal copies of $[[\,deptName\,]]$ and $[[\,brName\,]]$. Again, we have that any two almost-copies of PD in PD′ will contain $P_1'|_{[[\,deptName\ brName\,]]}$ or $P_2'|_{[[\,deptName\ brName\,]]}$. Therefore any two almost-copies of PD in PD′ will contain value-equal copies of $[[\,deptName\,]]$ and $[[\,brName\,]]$, so that PD_XFD4 is satisfied. □

As for the RDM, we say that an XML data tree T' satisfies a given set Σ of XFDs, denoted by $\models_{T'} \Sigma$, if T' satisfies each XFD in Σ. Satisfaction of a given set of XFDs by an XML data tree usually implies the satisfaction of other XFDs. The notions of implication and derivability (with respect to a rule system $\mathcal{R}$) are defined analogously to similar notions in the RDM.

Let Σ be a set of XFDs and $X \to Y$ a single XFD. If $X \to Y$ is satisfied in every XML data tree which satisfies Σ, then Σ *implies* $X \to Y$, written as $\Sigma \models X \to Y$. The *semantic closure of* Σ, denoted by Σ^*, is the set of all XFDs which are implied by Σ, that is, $\Sigma^* = \{X \to Y \mid \Sigma \models X \to Y\}$.

Given a rule system $\mathcal{R}$, we call an XFD $X \to Y$ *derivable* from Σ by $\mathcal{R}$, denoted by $\Sigma \vdash_{\mathcal{R}} X \to Y$, if there is a finite sequence of XFDs, whose last element is $X \to Y$, such that each XFD in the sequence is in Σ or can be obtained from Σ by applying one of the inference rules in $\mathcal{R}$ to a finite number of previous

XFDs in the sequence. The *syntactic closure of Σ with respect to the rule system* $\mathcal{R}$, denoted $\Sigma^+_{\mathcal{R}}$, is the set of all XFDs which are derivable from Σ by means of inference rules in $\mathcal{R}$, that is, $\Sigma^+_{\mathcal{R}} = \{X \to Y \mid \Sigma \vdash_{\mathcal{R}} X \to Y\}$. Whenever the rule system is clearly understood, we may omit $\mathcal{R}$.

An inference rule is called *sound* if for any given set Σ of XFDs, every XFD which may be derived from Σ due to that rule is also implied by Σ. A rule system $\mathcal{R}$ is *sound* if all inference rules in $\mathcal{R}$ are sound. In other words, $\mathcal{R}$ is sound if every XFD which is derivable from Σ by $\mathcal{R}$ is also implied by Σ (i.e. $\Sigma^+_{\mathcal{R}} \subseteq \Sigma^*$). A rule system is said to be *complete* if it is possible to derive every XFD which is implied by Σ (i.e. $\Sigma^* \subseteq \Sigma^+_{\mathcal{R}}$).

5 Sound Inference Rules

In this section, we assemble sound inference rules that yield a sound and complete rule system as we will demonstrate later on.

Lemma 11. *Let T be an XML schema tree, and let X, Y, W, Z be r_T-subgraphs of T. The following inference rules for XFDs are sound:*

$$\text{(union rule)} \quad \frac{X \to Y, X \to Z}{X \to Y \cup Z}$$

$$\text{(reflexivity axiom)} \quad \frac{}{X \to Y} \quad Y \text{ is an } r_T\text{-subgraph of } X$$

$$\text{(subtree rule)} \quad \frac{X \to Y}{X \to Z} \quad Z \text{ is an } r_T\text{-subgraph of } Y$$

$$\text{(supertree rule)} \quad \frac{W \to Y}{X \to Y} \quad W \text{ is an } r_T\text{-subgraph of } X$$

For an XML schema tree T, let R_T denote the union of all r_T-walks of $T_{\leq 1}$.

Lemma 12. *Let T be an XML schema tree, and let X be an r_T-subgraph of T. The following inference rule is sound for XFDs:*

$$\text{(root axiom)} \quad \frac{}{X \to R_T}$$

Surprisingly, the transitivity rule from the RDM does not hold for XML in the presence of frequencies. Consider the XFDs $X \to Y$ and $Y \to Z$ defined on some XML schema tree T. For an XML data tree $T' \rhd T$, if any two almost-copies of T are missing a copy of some r_T-walk of Y, then T' trivially satisfies $Y \to Z$. Therefore it would be possible for two almost-copies to be not value-equal on Z while being value-equal on and not missing a copy of X, that is, $X \to Z$ can be violated.

However we can define a restricted form of the transitivity rule which is sound for the derivation of XFDs. The main idea behind such an inference rule is to use frequencies to ensure that two almost-copies are not missing a copy of every

r_T-walk of the middle term Y whenever they are not missing a copy of every r_T-walk of X or Z. The notion of Y being X, Z-compliant in the following definition accomplishes this.

Definition 13. *Let X, Y, Z be r_T-subgraphs in an XML schema tree T. We say Y* is X, Z-compliant *if and only if $Y \subseteq (X \cup C) \cup T_{\geq 1}$ for each r_T-walk C of Z.*

Example 14. In the XML schema tree PD in Figure 2, it is easy to see that $[[\,brName\,]] \subseteq ([[\,deptName\,]] \cup [[\,brLocation\,]]) \cup \mathrm{PD}_{\geq 1}$ holds, that is, $[[\,brName\,]]$ is $[[\,deptName\,]]$,$[[\,brLocation\,]]$-compliant. □

Lemma 15. *Let T be an XML schema tree, and let X, Y, Z be r_T-subgraphs of T. The following inference rule is sound for XFDs:*

$$\text{(restricted-transitivity rule)} \qquad \frac{X \to Y, Y \to Z}{X \to Z} \quad Y \text{ is } X, Z\text{-compliant}$$

Example 16. Recall the XML schema tree PD in Figure 2 and all XFDs specified in Example 10. There are two r_{PD}-walks in $\mathrm{PD}_{\leq 1}$, yielding $R_{\mathrm{PD}} = [[\,address\ phone\,]]$. Using the *root axiom* we derive XFDs like $[[\,empID\,]] \to [[\,address\ phone\,]]$ (let this be denoted by PD_XFD6) and $[[\,empName\ brName\,]] \to [[\,address\ phone\,]]$.

The *supertree rule* enables us to derive from PD_XFD6 the XFD $[[\,empID\ empName\ brName\,]] \to [[\,address\ phone\,]]$. Applying the *subtree rule* to PD_XFD6 gives us the XFDs $[[\,empID\,]] \to [[\,address\,]]$ and $[[\,empID\,]] \to [[\,phone\,]]$. Using the *reflexivity axiom*, we can derive the XFD $[[\,empID\ empName\ brName\,]] \to [[\,empID\ empName\,]]$. From PD_XFD1 and PD_XFD2 and an application of the *union rule* we obtain the XFD $[[\,brName\,]] \to [[\,brLocation\ brPhone\,]]$.

Since $[[\,brName\,]]$ is $[[\,deptName\,]]$,$[[\,brLocation\,]]$-compliant, an application of the *restricted-transitivity rule* to PD_XFD4 and PD_XFD1 yields the XFD $[[\,deptName\,]] \to [[\,brLocation\,]]$. □

Next, we define the notion of a unit of some r_T-walk which is needed for the final inference rule presented in this section.

Definition 17. *Let B be an r_T-walk of some XML schema tree T. The* unit of B, *denoted by U_B, is the union of all r_T-walks sharing some */+-arc with B.*

We continue with some useful observations about the unit of an r_T-walk. For one, it is the case that $U_C = U_B$ for any r_T-walk $C \in U_B$. Furthermore, in any data tree $T' \rhd T$, every almost-copy of $T - U_B$ together with any almost-copy of U_B form an almost-copy of T in T'. In particular, for any two almost-copies T'_1, T'_2 of T in T', it is the case that $T'_1|_{T-U_B} \cup T'_2|_{U_B}$ and $T'_2|_{T-U_B} \cup T'_1|_{U_B}$ are also almost-copies of T in T'. The mix-and-match approach is only possible because $T'_2|_{U_B}$ shares with $T'_1|_{T-U_B}$ exactly those arcs (and vertices) which $T'_2|_{U_B}$ shares with $T'_2|_{T-U_B}$, and likewise $T'_1|_{U_B}$ shares with $T'_2|_{T-U_B}$ exactly those arcs which $T'_1|_{U_B}$ shares with $T'_1|_{T-U_B}$.

Lemma 18. *Let T be an XML schema tree, let X be an r_T-subgraph of T, and let B an r_T-walk of T. The following inference rule is sound for XFDs:*

$$\text{(noname rule)} \qquad \frac{\big((X \cup B) \cup T_{\geq 1} - U_B\big) \cup X \to B}{X \to B}$$

Example 19. The noname rule allows us to derive XFDs which have not been derivable using the other derivation rules only, e.g., the new XFD $[[\,empName\,]] \to [[\,brName\,]]$ for our example above. To see this, we first find $([[\,empName\,]] \cup [[\,brName\,]]) \cup \mathrm{PD}_{\geq 1}$ as the r_{PD}-subgraph $[[\,empID\ empName\ deptName\ brName\,]]$ of PD, and the unit $U_{[[\,brName\,]]}$ as the r_{PD}-subgraph $[[\,brName\ brLocation\ brPhone\,]]$. This amounts to the premise of the noname rule being the XFD $[[\,empID\ empName\ deptName\,]] \to [[\,brName\,]]$, which can be derived from $[[\,deptName\,]] \to [[\,brName\,]]$ (that is, PD_XFD4) by means of the supertree rule. □

6 A Sound and Complete Rule System

In this section, we observe that the inference rules assembled above form a complete rule system for XFDs in the presence of frequencies. Let the *$\mathcal{F}$-rule system* consist of the following inference rules: *reflexivity axiom, root axiom, subtree rule, supertree rule, union rule, restricted-transitivity rule* and *noname rule.*

We take the usual approach to verifying completeness. Consider an XML schema tree T and a set Σ of XFDs on T. If $X \to Y$ cannot be derived from Σ by means of the inference rules, then we show that there is an XML data tree $T' \rhd T$ such that $\models_{T'} \Sigma$ but $\not\models_{T'} X \to Y$. Because of the union rule, this means that there is some r_T-walk $B \in Y$ such that $X \to B$ is not derivable from Σ and T' does not satisfy $X \to B$. Therefore T' must contain two almost-copies T'_1, T'_2 of T such that $T'_1|_X = T'_2|_X \cong X$ and $T'_1|_B \neq T'_2|_B$. In the sequel, we will outline a *general construction* for such a *counterexample data tree* T'.

Without frequencies, we can construct a counterexample data tree from the arc-disjoint union of exactly two copies T'_a, T'_b of $X \cup B$ that are value-equal only on X. Particularly, T'_a, T'_b are both missing a copy of every r_T-walk not in $X \cup B$. In the presence of frequencies, however, we face the additional complication that at least one almost-copy of T in T' must contain a copy of $(X \cup B) \cup T_{\geq 1}$. This means, in addition to $X \cup B$, we also need to determine whether or not T'_1 and T'_2 should be value-equal on any of the remaining r_T-walks in $(X \cup B) \cup T_{\geq 1}$, keeping in mind that T' must still satisfy Σ.

We first define the analogous of the closure of a set of attributes in the RDM.

Definition 20. *Let T be an XML schema tree, X be an r_T-subgraph, and Σ be a set of XFDs on T. Further let $\mathcal{R}$ be a rule system. The* pre-closure $X^+_{\mathcal{R}}$ *of X with respect to Σ and $\mathcal{R}$ is the following r_T-subgraph of T:*

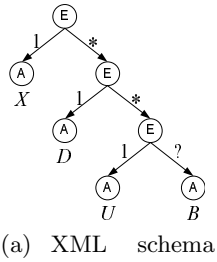
$$X^+_{\mathcal{R}} = \bigcup \{Y \mid X \to Y \in \Sigma^+_{\mathcal{R}}\}$$

As pointed out above, we need two almost-copies T'_1, T'_2 of T such that $T'_1|_X = T'_2|_X \cong X$ and $T'_1|_B \neq T'_2|_B$. To ensure $\models_{T'} \Sigma$ we must further guarantee that $T'_1|_{X^+_{\mathcal{F}}} = T'_2|_{X^+_{\mathcal{F}}}$. For that, the two almost-copies of T must not be missing value-equal copies of $X^+_{\mathcal{F}} \cap ((X \cup B) \cup T_{\geq 1})$. A peculiar situation is encountered: there may be XFDs that are not implied by Σ (and hence not derivable by the $\mathcal{F}$-rule system), but need to be non-trivially satisfied in this situation because at least one of T'_1, T'_2 is not missing a copy of $(X \cup B) \cup T_{\geq 1}$. The restricted-transitivity rule yields $(X^+_{\mathcal{F}})^+_{\mathcal{F}} \supseteq X^+_{\mathcal{F}}$, but $X^+_{\mathcal{F}}$ will in general not be a closure as its counterpart attribute closure in the RDM.

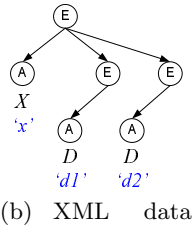
Example 21. Our previous observation is illustrated by the XML schema tree S in Figure 5(a) and the set Σ of XFDs defined on S. We find that $[[U]] \not\subseteq ([[X]] \cup [[D]]) \cup S_{\geq 1}$, that is, $[[U]]$ is not $[[X]], [[D]]$-compliant. Therefore, we cannot use the restricted-transitivity rule to derive $[[X]] \to [[D]]$. In fact, the $\mathcal{F}$-rule does not allow us to derive $[[X]] \to [[D]]$. If an XML data tree compatible with S should satisfy Σ and violate $[[X]] \to [[D]]$ it must simply be missing a copy of $[[U]]$, see Figure 5(b).

However, if an XML data tree compatible with S is not missing a copy of $[[B]]$, then it will not be missing a copy of $[[U]]$ either. Consider the data tree S'' in Figure 5(c). It contains exactly two almost-copies of S, which we denote by S'_1, S'_2. Since $\models_{S''} \Sigma$, we have $\models_{S''} [[X]] \to [[U]]$ and $\models_{S''} [[U]] \to [[D]]$. It follows from $\models_{S''} [[X]] \to [[U]]$ and $S'_1|_{[[X]]} = S'_2|_{[[X]]} \cong [[X]]$ that $S'_1|_{[[U]]} = S'_2|_{[[U]]}$. Moreover, since $[[U]] \in ([[X]] \cup [[B]]) \cup S''_{\geq 1}$, neither of S'_1, S'_2 is missing a copy of $[[U]]$. This means we have $S'_1|_{[[U]]} = S'_2|_{[[U]]} \cong [[U]]$, and hence $S'_1|_{[[D]]} = S'_2|_{[[D]]}$ due to $\models_{S''} [[U]] \to [[D]]$. □

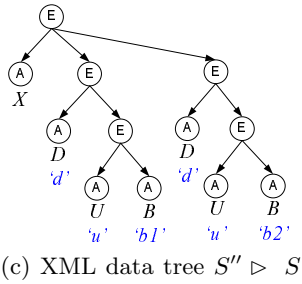
Consequently, it is insufficient to stop after having considered only $X^+_{\mathcal{F}}$. Since T'_1, T'_2 are value-equal on and not missing a copy of $X^+_{\mathcal{F}} \cap ((X \cup B) \cup T_{\geq 1})$, it follows from $\models_{T'} \Sigma$ that T'_1, T'_2 must be value-equal on and not missing a copy of $(X^+_{\mathcal{F}} \cap ((X \cup B) \cup T_{\geq 1}))^+_{\mathcal{F}} \cap ((X \cup B) \cup T_{\geq 1})$. This then



(a) XML schema tree S. Let $\Sigma = \{[[X]] \to [[U]], [[U]] \to [[D]]\}$ be a set of XFDs on S.



(b) XML data tree $S' \rhd S$ with $\models_{S'} \Sigma$ showing $[[X]] \to [[D]]$ can be violated.



(c) XML data tree $S'' \rhd S$ with $\models_{S''} \Sigma$, where S'' is not missing a copy of $[[B]]$. It is not possible that $\not\models_{S''} [[X]] \to [[D]]$.

Fig. 5. XML schema tree and data trees illustrating that there can be an XFD which is not derivable but which is satisfied whenever there occur copies of certain r_S-walks

means that T_1', T_2' must be value-equal on and not missing a copy of $((X_{\mathcal{F}}^{+} \cap ((X \cup B) \cup T_{\geq 1}))_{\mathcal{F}}^{+} \cap ((X \cup B) \cup T_{\geq 1}))_{\mathcal{F}}^{+} \cap ((X \cup B) \cup T_{\geq 1})$, and so on. In the process, we obtain a sequence of pre-closures restricted to $(X \cup B) \cup T_{\geq 1}$. Eventually there is some fix-point X_n since XML schema trees are finite so that there are only finitely many r_T-walks in $(X \cup B) \cup T_{\geq 1}$. In summary, T_1', T_2' must be value-equal on and not missing a copy of the r_T-subgraph X_n in order for $\models_{T'} \Sigma$.

This outlines our general approach for constructing a counterexample data tree, though the actual proof of completeness uses some additional considerations. Recall that $T_1'|_{T-U_B} \cup T_2'|_{U_B}$ and $T_2'|_{T-U_B} \cup T_1'|_{U_B}$ are possibly further almost-copies of T in T'. In particular, if $T_1'|_{T-U_B} \neq T_2'|_{T-U_B}$ and $T_1'|_{U_B} \neq T_2'|_{U_B}$ then there are at least four almost-copies of T in T'. To simplify the discussion it is desirable to have only two almost-copies of T in the counterexample data tree under construction. To ensure this, we can force T' to contain only one copy of $T|_{T-U_B} \cap ((X \cup B) \cup T_{\geq 1})$. Actually, $T|_{T-U_B} \cap ((X \cup B) \cup T_{\geq 1})$ equals $(X \cup B) \cup T_{\geq 1} - U_B$. Therefore, we want to have $T_1'|_{(X \cup B) \cup T_{\geq 1} - U_B} = T_2'|_{(X \cup B) \cup T_{\geq 1} - U_B}$.

However, there might be some r_T-walk in $(X \cup B) \cup T_{\geq 1} - U_B$ that is not in the r_T-subgraph X_n introduced above. This is rectified by actually computing the sequence of restricted pre-closures starting with $X_0 = ((X \cup B) \cup T_{\geq 1} - U_B) \cup X$ rather than just X. The noname rule guarantees that each restricted pre-closure in the sequence remains unaffected except for the additional r_T-subgraph $X_0 - X$.

Example 22. We demonstrate the approach described thus far with an example. Consider the XML schema tree Q shown in Figure 6. (Note that we left out some vertex names in Q for convenience.) Let $\Sigma = \{[[DB]] \to [[C]], [[X]] \to [[B]]\}$ be a set of XFDs given on Q. It is easy to check that $[[XF]] \to [[W]]$ is not derivable from Σ by the $\mathcal{F}$-rule system.

Suppose we want to construct a counterexample data tree $Q' \rhd Q$ such that $\models_{Q'} \Sigma$ but $\not\models_{Q'} [[XF]] \to [[W]]$. The r_Q-subgraph $([[XF]] \cup [[W]]) \cup Q_{\geq 1}$ is

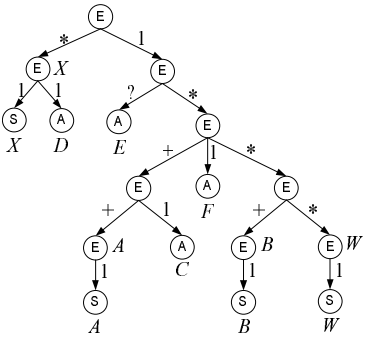


Fig. 6. An XML schema tree Q

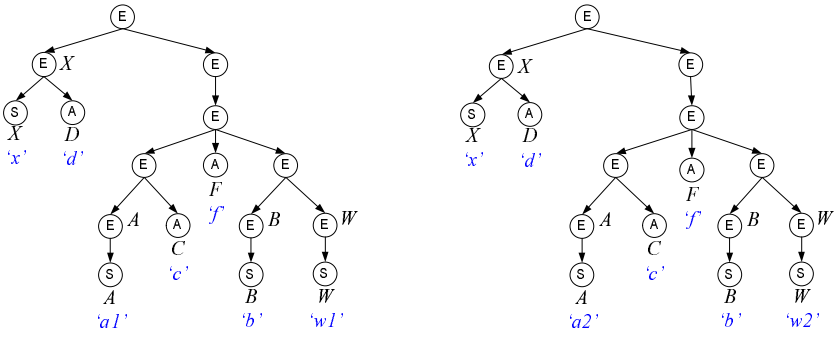


Fig. 7. Two almost-copies Q'_1, Q'_2 of the XML schema tree Q

the schema tree Q without the r_Q-walk $[[\,E\,]]$. Thus, each restricted pre-closure must not contain $[[\,E\,]]$. The sequence of restricted pre-closures is as follows:

$$\begin{aligned}
[[\,XF\,]]_0 &= \big(([[\,XF\,]] \cup [[\,W\,]]) \cup Q_{\geq 1} - U_{[[\,W\,]]}\big) \cup [[\,XF\,]] \\
&= ([[\,XFABCDW\,]] - [[\,ABCFW\,]]) \cup [[\,XF\,]] \\
&= [[\,XD\,]] \cup [[\,XF\,]] = [[\,XFD\,]] \\
[[\,XF\,]]_1 &= ([[\,XFD\,]])^{+}_{\mathcal{F}} \cap \big(([[\,XF\,]] \cup [[\,W\,]]) \cup Q_{\geq 1}\big) = [[\,XFDB\,]] \\
[[\,XF\,]]_2 &= ([[\,XFDB\,]])^{+}_{\mathcal{F}} \cap \big(([[\,XF\,]] \cup [[\,W\,]]) \cup Q_{\geq 1}\big) = [[\,XFDBC\,]] \\
&= [[\,XF\,]]_3 = [[\,XF\,]]_4 = \ldots
\end{aligned}$$

Therefore, Q' should contain two almost-copies Q'_1, Q'_2 of Q which are value-equal on and not missing a copy of the r_Q-subgraph $[[\,XFDBC\,]]$. Two such almost-copies of Q are depicted in Figure 7. □

As discussed above, we construct T' in general by taking the union of two copies of $(X \cup B) \cup T_{\geq 1}$ that are value-equal only on X_n. Of course, T' should be compatible with T and thus conform to the frequencies specified on T. Consequently, in T' the two copies may need to share certain arcs or even copies of entire r_T-walks of T. We next provide a notion for describing that two almost-copies of some schema tree T share the same copy of some r_T-walk of T in a compatible data tree T'.

Definition 23. *Let T be an XML schema tree. Two almost-copies T'_1, T'_2 of T in an XML data tree $T' \rhd T$ are said to* coincide on an r_T-walk B *of T if and only if $T'_1|_B$ and $T'_2|_B$ are graph unions of exactly the same set of arcs in T'. Two almost-copies of T* coincide on an r_T-subgraph Y *of T if and only if they coincide on each r_T-walk of Y.*

Example 24. Recall the XML data tree PD′ in Figure 3 containing almost-copies P'_1 to P'_4 of PD, cf. Figure 4. P'_1 and P'_3 coincide on the r_{PD}-subgraph $[[\,empID\ deptName\,]]$ but do not coincide on any other r_{PD}-walk. □

The following amalgamation operator allows us to specify how two almost-copies of an XML schema tree can be combined to construct an XML data tree.

Definition 25. *Let T be an XML schema tree, T'_a, T'_b be two almost-copies of T, and X be an r_T-subgraph of T. The* amalgamation of T'_a and T'_b on X, *denoted $T'_a \amalg_{[X]} T'_b$, is the XML data tree obtained by taking the graph union of T'_a and T'_b in such a way that T'_a and T'_b coincide on X, and T'_a and T'_b only share all arcs in their projections to $X \cup T_{\leq 1}$.*

As mentioned before, we construct the counterexample data tree by amalgamating two copies T'_a, T'_b of $(X \cup B) \cup T_{\geq 1}$ that are value-equal only on X_n. By construction, both T'_a, T'_b are almost-copies of T in T'. Recall, however, that $T'_a|_{(X\cup B)\cup T_{\geq 1} - U_B} \cup T'_b|_{U_B}$ and $T'_b|_{(X\cup B)\cup T_{\geq 1} - U_B} \cup T'_a|_{U_B}$ are possibly further almost-copies of T in an XML data tree containing T'_a and T'_b. From $X_0 = ((X \cup B) \cup T_{\geq 1} - U_B) \cup X \subseteq X_n$ we know that T'_a and T'_b are value-equal on $(X \cup B) \cup T_{\geq 1} - U_B$. In order for the constructed data tree T' to contain only two almost-copies of T (namely T'_a and T'_b), we require that T'_a and T'_b are amalgamated on $(X \cup B) \cup T_{\geq 1} - U_B$. The resulting XML data tree T' is in fact the desired counterexample data tree we are looking for.

As a consequence, we conclude the completeness of the $\mathcal{F}$-rule system.

Theorem 26. *The $\mathcal{F}$-rule system is sound and complete for XFDs in the presence of frequencies.*

Example 27. Let us continue with Example 22. To construct a counterexample data tree $Q' \rhd Q$ such that $\models_{Q'} \Sigma$ but $\not\models_{Q'} [[XF]] \to [[W]]$, we use two copies Q'_a, Q'_b of $([[XF]] \cup [[W]]) \cup Q_{\geq 1} = [[XFABCDW]]$ that are value-equal on the r_Q-subgraph $[[XFDBC]]$. Incidentally, these can be the two almost-copies Q'_1, Q'_2 of Q shown in Figure 7.

From Example 22, we have that $([[XF]] \cup [[W]]) \cup Q_{\geq 1} - U_{[[W]]} = [[XD]]$. As suggested, we amalgamate Q'_a and Q'_b on $[[XD]]$. The resulting XML data tree $Q'_a \amalg_{[[XD]]} Q'_b$, denoted Q' for short, is shown in Figure 8. Note that Q' contains only the two almost-copies Q'_1, Q'_2 of Q. Though we have left out the names of some vertices for convenience, the homomorphism between Q' and

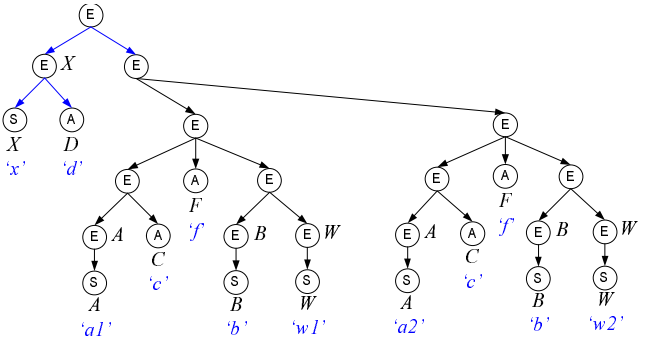


Fig. 8. The XML data tree $Q' = Q'_a \amalg_{[XD]} Q'_b$ where $Q' \rhd Q$. Here Q'_a and Q'_b correspond to the almost-copies Q'_1 and Q'_2 in Figure 7.

Q is the name-preserving mapping which maps every vertex from Q' to the vertex carrying the same name in Q. By examining the frequencies given on Q in Figure 6, it can easily be verified that $Q' \rhd Q$. Thus Q' is the desired counterexample data tree. □

7 The Impact of Identifiers

Elements in an XML document can be identified using ID-attributes. The values of ID-attributes have to be unique throughout the document. This feature is borrowed from object-oriented data models [15, 23] and is supported, e.g., by DTDs. (XSDs also support ID-attributes, but provide further opportunities to declare unique values, too.) In this section, we discuss XFDs in the presence of XML schema trees where attributes may be declared as ID-attributes. We first revise some earlier definitions to account for the presence of ID-attributes.

We use "I" to differentiate an ID-attribute from an ordinary attribute. The kind assignment is thus extended to $kind : V_G \to \{E, A, I, S\}$. The possible frequencies for an arc $a = (v, w)$ where $kind(w) = I$ is the same as for $kind(w) = A$, that is, $freq(a) =?$ or 1. Furthermore, we call an r_T-walk to some ID-attribute an *identifier*. Let V_G^I consist of all vertices in V_G of kind I. Leaves are now vertices of kind A, I or S, that is, $L_G = V_G^A \cup V_G^S \cup V_G^I$. It is not necessary to modify the definitions of XML data tree and equivalence of XML data trees.

Values assigned to ID-attributes need to be unique in an XML data tree T', that is, for every leaf $v \in V_{T'}^I$ of the data tree T' there does not exist any other leaf $u \in L_{T'}$ such that $val(v) = val(u)$. We refer to this as the *unique identifier value constraint*. Now, the revised version of Definition 8 [compatible] simply requires that T' also satisfies the unique identifier value constraint.

It is easy to see that all inference rules which are sound in the presence of frequencies are also sound in the presence of frequencies and identifiers. However, a few more inference rules are required to obtain a sound and complete rule system again. The first new rule stems from the uniqueness of values taken by ID-attributes. For an r_T-subgraph X, let X_{ID} be the union of all identifiers in X, that is, the r_T-subgraph of X whose leaves are just the leaves of kind I of X.

Lemma 28. *Let T be an XML schema graph, and X be an r_T-subgraph of T with X_{ID} being non-empty. The following inference rule is sound for XFDs:*

$$\text{(identifier axiom)} \qquad \frac{}{X_{ID} \to X_{ID} \cup T_{\leq 1}}$$

If we allow the left hand side of an XFD to be an empty r_T-subgraph, then we can rewrite the root axiom from Lemma 12 such that the identifier axiom is a straightforward generalisation of the rewritten root axiom.

We next extend the notion of a unit of some r_T-walk to account for the presence of identifiers. This will then provide us a generalisation of the noname rule from Lemma 18.

Definition 29. *Let X be an r_T-subgraph, and B be an r_T-walk of an XML schema graph T. The* unit of B relative to X, *denoted by U_B^X, is the union of all r_T-walks sharing some */+-arc a with B where a is not an arc of X_{ID}.*

Analogous to U_B, for any r_T-walk $C \in U_B^X$, we find that $U_C^X = U_B^X$. It remains to check whether for any two almost-copies T_1', T_2' of T in an XML data tree $T' \rhd T$, it is still true that $T_1'|_{T-U_B^X} \cup T_2'|_{U_B^X}$ and $T_2'|_{T-U_B^X} \cup T_1'|_{U_B^X}$ are almost-copies of T in T'.

If X_{ID} is empty, that is, there is no identifier in X, then $U_B^X = U_B$. Obviously in this case, $T_1'|_{T-U_B^X} \cup T_2'|_{U_B^X}$ and $T_2'|_{T-U_B^X} \cup T_1'|_{U_B^X}$ are almost-copies of T from the observation we made before. Suppose instead that X_{ID} is non-empty. The above observation is not true unless an additional condition is satisfied. Let $I \in X_{ID}$ be an identifier which shares with each r_T-walk of $U_B - U_B^X$ the last */+-arc which that r_T-walk shares with U_B^X. For any two almost-copies T_1', T_2' of T such that $T_1'|_I = T_2'|_I \cong I$, we can then observe that $T_1'|_{T-U_B^X} \cup T_2'|_{U_B^X}$ and $T_2'|_{T-U_B^X} \cup T_1'|_{U_B^X}$ are also almost-copies of T. Similar to previously, this mix-and-match is possible because the unique identifier value constraint guarantees that $T_2'|_{U_B^X}$ shares with $T_1'|_{T-U_B^X}$ exactly those arcs (and vertices) which $T_2'|_{U_B^X}$ shares with $T_2'|_{T-U_B^X}$, while $T_1'|_{U_B^X}$ shares with $T_2'|_{T-U_B^X}$ exactly those arcs which $T_1'|_{U_B^X}$ shares with $T_1'|_{T-U_B^X}$.

Lemma 30. *Let T be an XML schema graph, X be an r_T-subgraph of T, and B be an r_T-walk of T. The following inference rule is sound for XFDs:*

$$\text{(generalised noname rule)} \qquad \frac{\left((X \cup B) \cup T_{\geq 1} - U_B^X\right) \cup X \to B}{X \to B}$$

Note that, if X_{ID} is a empty, the generalised noname rule reduces to the noname rule from Lemma 18.

We may finally record the main result of this paper.

Theorem 31. *The $\mathcal{I}$-rule system consisting of the reflexivity axiom, root axiom, subtree rule, supertree rule, union rule, restricted-transitivity rule, identifier axiom and generalised noname rule is sound and complete for XFDs in the presence of frequencies and identifiers.*

8 Discussion

In this section, we assemble some remarks on related work, and point out similarities and differences.

8.1 Relational FDs in the Presence of Null Values

The observation that the transitivity rule is no longer sound for XML corresponds to a similar observation for relational databases in the presence of missing information, cf. Lien [12], who also provides an axiomatisation for functional

dependencies in relational databases with null values. Later Atzeni and Morfuni [6] generalised this axiomatisation to functional dependencies in the presence of null values (NFDs) and existence constraints (ECs) over flat relations. An existence constraint is an expression of the form $e : X \vdash Y$ which is satisfied by a relation if each tuple t which is X-total (i.e. $t[X]$ does not contain any null values) is also Y-total. Like frequencies, ECs offer a means to control the presence of missing values. In particular, ECs are required for defining a restricted form of the transitivity rule which is sound in the presence of null values. ECs may express some, but not all, frequencies and conversely frequencies may express some, but not all, ECs. It should be noted that the Atzeni-Morfuni rule system contains additional inference rules for the derivation of ECs which are not required for frequencies. More importantly, the $\mathcal{F}$-rule system contains two additional inference rules to the Atzeni-Morfuni rule system, namely the root axiom and the noname rule, which arise from the nested structure of XML data.

8.2 XML Schema Trees and DTDs

The currently most popular approach towards functional dependencies for XML is due to Arenas and Libkin [3, 4], who define XFDs in the presence of a DTD. In practice, XML documents do not always possess a DTD, but may be developed either from scratch or using some other specification (such as an XSD). Therefore, we use schema graphs rather than DTDs for the definition of functional dependencies. However, if a DTD is available it may be used to derive an XML schema tree, cf. [9].

It should be noted that DTDs and XML schema graphs differ in their expressiveness. DTDs use regular expressions to specify the element type definition associated with a particular element name, that is, the permitted combinations of names for the children of an element of that name. In our approach, we do not consider disjunctions. If a DTD contains a disjunctive expression, we first try to rewrite it without using disjunction. For example, $(A|B|C)^*$ may be rewritten (A^*, B^*, C^*). Otherwise, we treat the disjunction as a sequential expression with each element name in the disjunction being assigned a frequency of * or ?. For example, if $(A|B|C^+)$ is a regular expression in a given DTD, then this is treated as $(A^?, B^?, C^*)$. This is similar to the idea of simplifying a DTD [14], and to considering simple regular expressions [4]. Recent empirical studies show that the vast majority of DTDs do not contain disjunctive expressions [7]. A regular expression s is *simple* if there is a non-disjunctive regular expression s' such that every word w in the language represented by s is a permutation of a word in the language represented by s', and vice versa. In [4], a DTD is called *simple* if it contains only simple regular expressions as element type definitions.

Secondly, regular expressions in DTDs may be recursive and thus give rise to an XML schema graph with cycles, or alternatively, a rational XML schema tree. In this paper, we restrict ourselves to finite schema trees which in turn correspond to a finite number of unfoldings of the recursive expression, cf. [10]. In [4], a DTD is called *non-recursive* if it contains no recursive regular expressions as element type definitions.

Thirdly, XML schema trees may contain different vertices with the same name, but different sets of children. DTDs do not allow element type definitions associated with a particular element name to be overwritten. This is a feature that XML schema trees share with tree automata, XSDs and Relax NG specifications of XML data. For a discussion, see [7].

8.3 XFDs Studied by Arenas and Libkin

For their definition of XFDs, Arenas and Libkin [3, 4] introduce the notion of a "tree tuple", where a tree tuple is a finite partial evaluation of the paths in the underlying DTD. An XFD which consists of a set of paths on the left hand side and a single path on the right hand side is said to be satisfied if whenever two tree tuples agree on the paths on the left hand side of an XFD they must also agree on the path on the right hand side of the XFD. Note that XFDs in [3, 4] may contain paths that end in an internal vertex, rather than in a leaf. In our approach, identifier-attributes are used instead to refer to a particular vertex. This is more in line with the original intentions of XML and provides more flexibility: identifier-attributes are specified if desired, but not compulsory.

If an XML schema tree T is obtained from a DTD, the almost-copies of T in an XML data tree T' conforming to T are just the tree representations of the maximal tree tuples over the DTD that are subsumed by T'. As a consequence of Theorem 26, we find that the $\mathcal{I}$-rule system is sound and complete for XFDs in the presence of a simple non-recursive DTD. While Arenas and Libkin do not give inference rules for their XFDs, they show that XFDs in the presence of a DTD cannot be finitely axiomatised. The proof given in [4] suggests that the usage of non-simple disjunctive expressions in DTDs causes XFDs not to be finitely axiomatisable. This is one reason for considering XFDs over XML schema trees without disjunctions. Furthermore, [4] also shows the implication problem for XFDs in the presence of a simple non-recursive DTD to be solvable in quadratic time. Note that finite axiomatisability is a stronger property than the existence of an algorithm as the former implies the latter but not the other way around [1].

8.4 XFDs Studied by Vincent and Liu

More recently, Vincent and Liu (partly together with co-authors) studied another kind of functional dependencies for XML [13, 18, 19, 20, 17, 21]. Their definition of XFDs (called *strong XFDs*) does not assume a DTD, but does also not consider frequencies. A major distinguishing feature of the approach taken by Vincent and Liu is the concept of strong satisfaction, cf. [5]. Similar to relations with null values, they suggest to think of an XML tree with missing information as representing a collection of "complete" XML trees with non-missing information. An XML data tree then strongly satisfies an XFD if every possible completion satisfies the XFD. Translated into our approach, an XML data tree containing at least one almost-copy which is not a copy of the given XML schema tree is regarded as being incomplete. That is, Vincent and Liu use the "unknown" interpretation (values exist but not currently known) for information deemed to

be missing. We remark that the "no information" interpretation is more general than the "unknown" interpretation as it also includes the "does not exist" interpretation (values do not exist). Due to the flexibility of XML data modelling, we argue that there is good reason for using the "no information" or "does not exist" interpretation of missing information, as done by our approach, and also by Arenas and Libkin [3, 4].

Vincent and Liu [19] provide a system of sound inference rules for strong XFDs which, however, is only proven to be complete for strong XFDs with a single path on the left hand side (called *unary XFDs*). Due to the different interpretation of null values, their system of inference rules looks quite different from ours, and also from the Armstrong rule system for the RDM. We also emphasise that the transitivity rule is sound for strong XFDs, but not sound for XFDs in our approach.

Finally, Liu, Vincent and Liu [13] also studied XFDs that use only paths ending in leaves (called *primary XFDs*) which is similar to our approach to XFDs, but without frequencies and identifiers. This paper does not address axiomatisation, but is focussed on the design of XML documents.

8.5 Further Approaches to XFDs

There is a number of further works that discuss functional dependencies for XML. Lee, Ling and Low [11] study XFDs for designing XML databases that are similar to the approach of Arenas and Libkin [3, 4], but they do not give a precise definition of XFDs. More recently, Wang and Topor [22] have defined XFDs that may use upward paths and are thus more expressive than other approaches, but do not address axiomatisation.

In [9, 10] we discuss several variations of XFDs in the presence of XML schema graphs. In particular, we introduce XFDs whose definition uses pre-images of v-subgraphs instead of almost-copies, and motivate their application by a number of examples. Moreover, we suggest to investigate axiomatisations for XFDs in the presence of XML schema graphs.

9 Conclusion

In this paper, we studied XFDs that are defined in the presence of an XML schema tree. An XFD is satisfied by an XML data tree if whenever two almost-copies of the schema tree coincide and are complete on the left hand side of the XFD then they must also coincide on the right hand side of the XFD. Our approach is similar to the approach by Arenas and Libkin [3, 4], but uses XML schema trees. This allows us to provide a sound and complete system of inference rules that may be used to solve the implication problem for the class of XFDs under discussion. Further, we extended our result to XFDs defined on XML schema trees that may contain ID-attributes. It should be mentioned, that our axiomatisation may be used to conclude a normal form for XML data that avoids redundancies caused by this class of XFDs, cf. [16].

References

1. Abiteboul, S., Hull, R., and Vianu, V. *Foundations of Databases.* Addison-Wesley, 1995.
2. Arenas, M., Fan, W., and Libkin, L. What's hard about XML schema constraints? In *Database and Expert Systems Applications - DEXA* (2002), vol. 2453 of *LNCS*, Springer, pp. 269–278.
3. Arenas, M., and Libkin, L. A normal form for XML documents. In *Principles of Database Systems - PODS* (2002), ACM, pp. 85–96.
4. Arenas, M., and Libkin, L. A normal form for XML documents. *ACM Trans. Database Syst. 29* (2004), 195–232.
5. Atzeni, P., and De Antonellis, V. *Relational database theory.* Benjamin-Cummings, 1993.
6. Atzeni, P., and Morfuni, N. M. Functional dependencies and constraints on null values in database relations. *Inform. and Control 70* (1986), 1–31.
7. Bex, G. J., Neven, F., and Van den Bussche, J. DTDs versus XML schema: A practical study. In *Web and Databases - WebDB* (2004), pp. 79–84.
8. Courcelle, B. Fundamental properties of infinite trees. *Theoret. Comput. Sci. 25* (1983), 95–169.
9. Hartmann, S., and Link, S. More functional dependencies for XML. In *Advances in Databases and Information Systems - ADBIS* (2003), vol. 2798 of *LNCS*, Springer, pp. 355–369.
10. Hartmann, S., Link, S., and Kirchberg, M. A subgraph-based approach towards functional dependencies for XML. In *Systemics, Cybernetics and Informatics - SCI* (2003), IIIS, pp. 200–205.
11. Lee, M.-L., Ling, T. W., and Low, W. L. Designing functional dependencies for XML. In *Advances in Database Technology - EDBT* (2002), vol. 2287 of *LNCS*, Springer, pp. 124–141.
12. Lien, Y. E. On the equivalence of database models. *J. ACM 29* (1982), 333–362.
13. Liu, J., Vincent, M., and Liu, C. Functional dependencies, from relational to XML. In *Perspectives of System Informatics - PSI* (2003), pp. 531–538.
14. Shanmugasundaram, J., Tufte, K., Zhang, C., He, G., DeWitt, D. J., and Naughton, J. F. Relational databases for querying XML documents: Limitations and opportunities. In *Very Large Databases - VLDB* (1999), pp. 302–314.
15. Thalheim, B. *Entity-relationship modeling.* Springer, Berlin, 2000.
16. Trinh, T. Functional dependencies for XML: Axiomatisation and normal form in the presence of frequencies and identifiers. MSc thesis, Massey University, 2004.
17. Vincent, M., and Liu, J. Completeness and decidability properties for functional dependencies in XML. *CoRR cs.DB/0301017* (2003), 1017.
18. Vincent, M., and Liu, J. Functional dependencies for XML. In *Asia-Pacific Web Conference - APWeb* (2003), vol. 2642 of *LNCS*, Springer, pp. 22–34.
19. Vincent, M., and Liu, J. Strong functional dependencies and a redundancy free normal form for XML. In *Systemics, Cybernetics and Informatics - SCI* (2003), IIIS, pp. 218–223.
20. Vincent, M., Liu, J., and Liu, C. Redundancy free mappings from relations to XML. In *Web-Age Information Management - WAIM* (2003), pp. 55–67.
21. Vincent, M., Liu, J., and Liu, C. Strong functional dependencies and their application to normal forms in XML. *ACM Trans. Database Syst. 29* (2004), 445–462.
22. Wang, J., and Topor, R. W. Removing xml data redundancies using functional and equality-generating dependencies. In *Database Technologies - ADC* (2005), vol. 39 of *CRPIT*, Australian Computer Society, pp. 65–74.
23. Weddell, G. E. Reasoning about functional dependencies generalized for semantic data models. *ACM Trans. Database Syst. 17* (1992), 32–64.