

# A subgraph-based approach towards functional dependencies for XML

Sven Hartmann, Sebastian Link and Markus Kirchberg  
Information Science Research Centre  
Massey University  
Palmerston North  
New Zealand  
{s.hartmann, s.link, m.kirchberg}@massey.ac.nz

## Abstract

*In this paper, we survey recent results on functional dependencies for XML. We present a subgraph-based approach towards functional dependencies in XML. The proposed framework allows us to define functional dependencies similar to those ones introduced in earlier works on this topic, but enables us to capture further kinds of functional dependencies which happen to be useful in designing XML documents. Within the paper we point out some essential differences between the various kinds of functional dependencies under discussion resulting in different expressiveness, different semantic behavior as well as different axiomatizations.*

## 1. Introduction

Along with the growing popularity of XML [7] as a markup language for publishing and exchanging data on the web, there is an increasing number of applications using native and persistent XML documents. The targets of modern information systems such as biological information systems are often too complex to be stored in relational tables. For XML provides a simple, universal and widely accepted standard for defining complex documents, there is an ongoing trend to use XML for representing and actually storing biological data, cf. [22].

It is not difficult to imagine that in the not-so-distant future persistent XML documents will be used to keep track of a large portion of newly generated molecular biology data as well as semi-structured data from other sources. Data that has to be conceptually modelled, organized, managed and analyzed. It must be emphasized that XML itself is merely a language for describing the syntax of a document, not its semantics. In relational databases, integrity constraints have been proven useful in making a good design, in preventing update anomalies, and in allowing the application of efficient methods for data storage, access and queries. Integrity constraints for XML have been addressed in [1, 20] as well as in a number of recent research papers [3, 8, 9, 12, 11] with focus on key constraints, inclusion constraints, inverse constraints, and path constraints.

Functional dependencies other than key dependencies have been little investigated for XML so far. As is true for relational databases [5, 6], inserting or modifying data

in the presence of functional dependencies are likely to be a major source of update anomalies due to redundant data in XML documents. In view of this observation it is rather astonishing that functional dependencies so far did not receive adequate attention in the literature on XML. In fact, there are only two papers [4, 16] addressing these constraints.

Before studying functional dependencies and actually using them in designing and managing XML documents, they have to be formally defined. Both [4, 16] study functional dependencies on the basis of paths evolving from the root in XML documents. This idea goes back to earlier research on functional dependencies in semantic and object-oriented data models [18, 21]. While this approach is a natural generalization of functional dependencies in relational databases, it is obviously not the only one. The difficulty with XML documents is that they incorporate far more structure than relational tables. As a consequence there are plenty of functional dependencies in XML documents which are not captured by previous works.

Our objective is to define functional dependencies on the basis of subgraphs in XML documents which allow us to specify functional dependencies similar to those suggested in [4, 16], but also a variety of further functional dependencies for XML which happen to appear quite naturally in practice. Of course, we do not claim the collection of definitions presented in this paper to be exhaustive, but a suitable starting point for further investigation.

The paper is organized as follows. In Sections 2–5 we provide preliminary notions such as XML graphs, XML schema graphs and XML data trees. In Section 6 we present a first definition of functional dependencies based on subgraphs of an XML schema graph and discuss properties such as inference rules. Sections 7 and 8 address additional features for XML schema graphs such as frequencies and identifiers. In Section 9 we extend the original definition of functional dependencies to further subgraphs of XML schema graphs, while Section 10 suggests an alternative definition of functional dependencies based on homomorphic pre-images rather than on copies of subgraphs.

## 2. XML graphs

We start with reviewing basic features of a simple XML tree model. We assume that the reader is familiar with standard notions from graph theory such as graphs, trees

and walks. For a comprehensive introduction to graph theory, we refer e.g. to [15]. All graphs within this paper are considered to be directed, without parallel arcs and finite unless stated otherwise. For every graph  $G$ , let  $V_G$  denote its vertex set and  $A_G$  its arc set.

A *rooted graph* is a graph  $G$  with one distinguished vertex  $r_G$ , called the *root* of  $G$ , such that every vertex can be reached from  $r$  by passing a directed path. In a rooted graph every vertex but  $r_G$  has at least one predecessor. For every vertex  $v$ , let  $Succ_G(v)$  denote its (possibly empty) set of successors in  $G$ . Vertices without successors are said to be *leaves*. Let  $L_G$  denote the set of leaves of  $G$ . Given a vertex  $v$  and a subset  $W \subseteq L_G$  of leaves, a *v-subgraph*  $H$  of  $G$  is the union of all directed walks from  $v$  to some  $w \in W$ . In particular, a *v-walk* of  $G$  is a directed walk from  $v$  to a single leaf  $w \in L_G$ . For any two  $v$ -subgraphs  $X$  and  $Y$  of  $G$ , let  $X \cup Y$  denote their union in  $G$ , which is again a  $v$ -subgraph of  $G$ . A *rooted tree* is a rooted graph  $T$  containing neither directed nor non-directed cycles. Clearly, every  $v$ -subgraph of  $T$  is a tree again.

Throughout, we suppose that there is a fixed set  $Names$  of names. An *XML graph* is a rooted graph  $G$  together with mappings  $name : V_G \rightarrow Names$  and  $kind : V_G \rightarrow \{E, A\}$  assigning every vertex its name and kind, respectively. Let  $V_G^E$  (and  $V_G^A$ ) consist of all vertices of kind  $E$  (or  $A$ , respectively) in  $V_G$ . We suppose that vertices of kind  $A$  are always leaves, while vertices of kind  $E$  can be either leaves or non-leaves. Further, we suppose that no vertex has two successors of kind  $A$  carrying the same name. If  $G$  is a rooted tree we also speak of an *XML tree*.

### 3. XML schema graphs

Graphs are frequently used to illustrate the structure of XML documents, cf. [1]. An *XML schema graph* is an XML graph  $G$  where no two successors of the same vertex have the same name and the same kind. Note that XML schema graphs should not be mixed up with the popular language XML SCHEMA [19] which can be used to describe XML documents. Rather, we use this term to emphasize the analogy with a database schema in traditional database design. The vertices of the XML schema graph represent types of elements and attributes used in an XML document. The kinds  $E$  and  $A$  tell us whether a vertex actually represents elements or attributes, respectively.

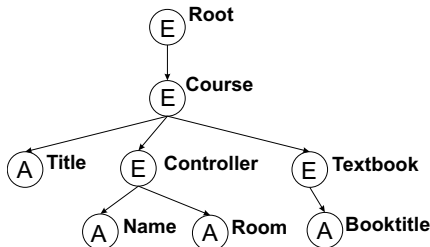


Fig. 1. An XML schema graph ...

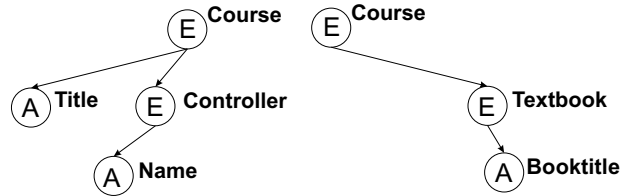


Fig. 2. ... and two  $v_{Course}$ -subgraphs  $X$  and  $Y$ .

An XML schema graph may easily be generated from a native XML document: For every element name (as well as for every attribute name)  $N$  we simply introduce a vertex  $v_N$  with  $name(v_N) = N$  and  $kind(v_N) = E$  (or  $A$ , respectively). Moreover, whenever an element (or attribute) of name  $N$  appears within some element of name  $M$ , then  $(v_M, v_N)$  should be an arc in the XML schema graph.

Earlier definitions of functional dependencies in XML [4, 16] assumed the existence of a DTD. In practice, however, XML documents do not always possess DTDs, but may be developed either from scratch or using some other description. Therefore, we use XML 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 graph, too. Due to the limited amount of space, we will not follow up this matter within the present paper.

### 4. Mappings between XML graphs

Let  $G'$  and  $G$  be two XML graphs, and consider a mapping  $\phi : V_{G'} \rightarrow V_G$ . We call  $\phi$  *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,  $\phi$  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  $\phi$  is a *homomorphism* between  $G'$  and  $G$  if the following conditions hold:

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

A homomorphism  $\phi$  between  $G'$  and  $G$  may be naturally extended to the arc set of  $G'$ : given an arc  $a' = (u', v')$  of  $G'$ ,  $\phi(a')$  denotes the arc  $(\phi(u'), \phi(v'))$  of  $G$ .

A homomorphism  $\phi$  between  $G'$  and  $G$  is an *isomorphism* if  $\phi$  is bijective and  $\phi^{-1}$  is an homomorphism, too. In this case,  $G'$  is said to be *isomorphic* to  $G$  or a *copy* of  $G$ . Similarly, a subgraph  $H'$  of  $G'$  is a copy of a subgraph  $H$  of  $G$  if the restriction of  $\phi$  to  $H'$  and  $H$  is an isomorphism between  $H'$  and  $H$ . If the homomorphism  $\phi$  between  $G'$  and  $G$  is not an isomorphism,  $G'$  may well contain several subgraphs which are copies of  $G$ . On the other hand,  $G'$  may well contain no subgraph at all which is a copy of  $G$ . An  $r_{G'}$ -subgraph  $H'$  of  $G'$  is a *subcopy* of  $G$  if it is the copy of some  $r_G$ -subgraph  $H$  of  $G$ , and an *almost-copy* of  $G$  if it is maximal with this property.

## 5. XML data trees

An XML data tree is an XML tree  $T$  together with an evaluation  $val : V_G^A \rightarrow STRING$  assigning every vertex  $v$  of kind  $A$  a (possibly empty) string  $val(v)$ . We call two isomorphic XML data trees  $T'$  and  $T$  equivalent if the isomorphism  $\phi : V_{T'} \rightarrow V_T$  between them is *evaluation-preserving*, that is,  $val(\phi(v')) = val(v')$  holds for every vertex  $v' \in V_{T'}^A$ .

Let  $T'$  be an XML data tree, and  $G$  be an XML schema graph. We call  $T'$  compatible with  $G$  if there is a homomorphism  $\phi : V_{T'} \rightarrow V_G$  between  $T'$  and  $G$ . Due to our definition of XML schema graphs this homomorphism is unique whenever it exists.

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, that is, by splitting vertices with more than one predecessor and thus removing its (directed as well as non-directed) cycles [10]. The resultant tree  $T_G$  is infinite (but still rational) or finite depending on whether  $G$  has directed cycles or not.

In the next section, when introducing functional dependencies, we always do this with respect to a fixed XML schema tree. This might be either the tree  $T_G$  (if it is finite) obtained from a given XML schema graph  $G$ , or some  $r_{T_G}$ -subgraph  $T$  of  $T_G$ . This approach is somewhat similar to the usage of tree tuples in [4]. Figures 3 and 4 show an XML schema graph  $G$  which is not a tree together with the XML schema tree  $T_G$  obtained by completely unfolding  $G$ .

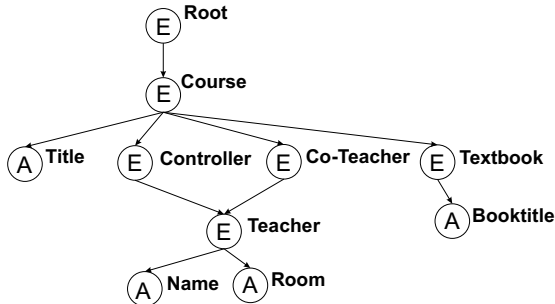


Fig. 3. An XML schema graph ...

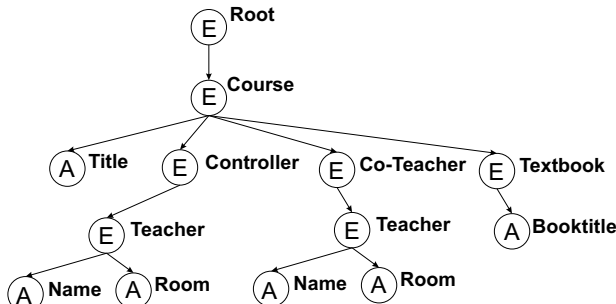


Fig. 4. ... and the unfolded XML schema tree.

## 6. Functional dependencies

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

**Definition 1** Given an XML schema tree  $T$ , a functional dependency (or XFD, for short) is an expression  $X \rightarrow Y$  where  $X$  and  $Y$  are  $r_T$ -subgraphs of  $T$ . Let  $T'$  be an XML data tree compatible with  $T$  and let  $\phi : V_{T'} \rightarrow V_T$  be the unique homomorphism between  $T'$  and  $T$ . Then  $T'$  satisfies the XFD under discussion 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 equivalent whenever the projections  $T'_1|_X$  and  $T'_2|_X$  are equivalent and copies of  $X$ .

Suppose, for example, whenever two courses run under the same title and are controlled by the same teacher they will use the same textbook. This may be expressed with the help of an XFD  $X \rightarrow Y$  where  $X$  and  $Y$  are the subgraphs in Figure 5, which are in fact  $r_T$ -subgraphs of the XML schema tree  $T$  in Figure 1. Moreover, Figure 8 below shows an XML data tree satisfying this functional dependency.

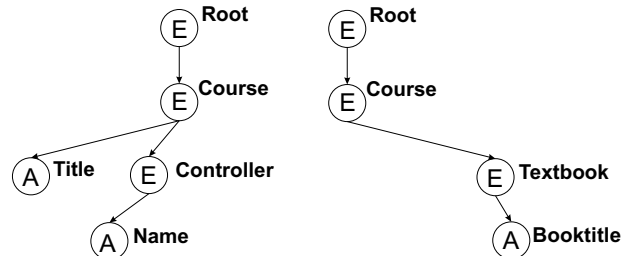


Fig. 5. Two  $v_{Root}$ -subgraphs  $X$  and  $Y$  of the XML schema tree in Figure 1.

As for ordinary database relations, the dependencies satisfied by an XML data tree are usually not independent. The notions of implication and derivability (with respect to some rule system  $\mathcal{R}$ ) can be defined analogously to the notions in the relational database model [2]. A single integrity constraint  $\sigma$  follows from a constraint set  $\Sigma$  if  $\sigma$  holds in every XML data tree which satisfies  $\Sigma$ . We also say that  $\Sigma$  implies  $\sigma$ . Of course, one will not inspect all possible XML data trees compatible with a given XML schema graph in order to decide the implications of a given set of functional dependencies. Rather, we are interested in inference rules which help to decide this question. An *inference rule* is an expression  $\frac{\Sigma'}{\sigma} \gamma$  where  $\Sigma'$  is a subset of  $\Sigma$ , and  $\gamma$  states some condition on  $\Sigma'$  which has to be satisfied if we want to apply this rule. If  $\Sigma$  contains a subset  $\Sigma'$  satisfying the condition  $\gamma$ , then  $\sigma$  may be *derived* from  $\Sigma$  due to that inference rule. An inference rule is *sound* if  $\Sigma$  implies every integrity constraint  $\sigma$  which may be derived from  $\Sigma$  due to that rule. The general problem is to find a rule system  $\mathcal{R}$  which allows us to derive every constraint  $\sigma$  which is implied by a given constraint set  $\Sigma$  within a fixed class of integrity constraints. Such a rule system is said to be *complete* for the implication of this class.

The following rules may easily be obtained as generalizations of similar inference rules for functional dependencies in the relational database model [5, 6]. We call them *reflexivity axiom*, *subtree rule*, *supertree rule*, *union rule*, *generalized union rule* and *transitivity rule*, respectively:

$$\frac{}{X \rightarrow Y} \text{ } Y \text{ is an } r_T\text{-subgraph of } X \quad (1)$$

$$\frac{X \rightarrow Y}{X \rightarrow Z} \text{ } Z \text{ is an } r_T\text{-subgraph of } Y \quad (2)$$

$$\frac{W \rightarrow Y}{X \rightarrow Y} \text{ } W \text{ is an } r_T\text{-subgraph of } X \quad (3)$$

$$\frac{X \rightarrow Y, X \rightarrow Z}{X \rightarrow Y \cup Z} \quad (4)$$

$$\frac{X \rightarrow Y, W \rightarrow Z}{X \cup W \rightarrow Y \cup Z} \quad (5)$$

$$\frac{X \rightarrow Y, Y \rightarrow Z}{X \rightarrow Z} \quad (6)$$

The reflexivity axiom, the subtree rule, the supertree rule, the union rule and the generalized union rule are sound for XFDs in Definition 1. Conversely, however, the transitivity rule is not sound in general. The latter observation is surprising as the transitivity rule is part of the well-known Armstrong system [5, 6] which is sound and complete for functional dependencies in the relational database model. On the other hand, however, the previous observation coincides with similar observations for relations with null-values indicating unavailable information, cf. [17].

Note that by Definition 1 the XFD in the example above is still valid if neither of two courses with the same title and same paper controller uses a textbook.

A further rule reflects the uniqueness of the root in XML documents. Given an XML schema tree  $T$  and a compatible XML data tree  $T'$  with homomorphism  $\phi : V_{T'} \rightarrow V_T$ , let  $R$  be the union of all those  $r_T$ -walks of  $T$  which do not share an arc with any other  $r_T$ -walk of  $T$  and whose single leaf is in  $V_T^E \cup (V_T^A \cap Succ_T(r_T))$ . It is easy to check that, any two almost-copies of  $R$  in  $T'$  are equivalent. In particular, if all leaves of  $R$  are of kind  $A$ , then  $G$  contains only one almost-copy of  $R$ . This gives rise to the *root axiom* for every  $r_T$ -subgraph  $X$ :

$$\frac{}{X \rightarrow R} \quad (7)$$

The reflexivity axiom, the subtree rule, the supertree rule, the union rule and the root axiom provide a sound and complete rule system for XFDs in Definition 1, cf. [14]. It is easy to see that the reflexivity axiom and the generalized union rule may be used to establish the supertree rule and the union rule. Hence, we may also use the generalized union rule to build a sound and complete rule system for XFDs in Definition 1.

## 7. Frequencies

So far, we allowed vertices in an XML data tree to possess an arbitrary number of successors of kind  $E$  sharing a certain name. This is quite natural if the XML schema graph  $G$  under discussion is obtained from a single XML

document. Consider some arc  $(v, w)$  in  $G$  where both  $v$  and  $w$  are of kind  $E$ . Even if each pre-image of  $v$  in an XML data tree  $T'$  has the same number of pre-images of  $w$  among its successors, we should not conclude that this is a must, unless additional semantic information is available. For example, DTDs and others provide means to restrict the number of occurrences of elements within another one. Similarly, attributes may be marked as '#required' or '#implied' indicating that their occurrence with every pre-image of some vertex  $v$  of  $G$  is mandatory or not. And even if we are not given a DTD, there are plenty of situations where occurrences of elements are bounded for some reasons. This observation motivates the investigation of XML schema graphs with frequencies assigned to its arcs.

Let  $G$  be an XML schema graph  $G$  together with a mapping  $freq : A_G \rightarrow \{*, ?, +, 1\}$ . We suppose that every arc  $a = (v, w)$  where  $w$  is of kind  $A$  has frequency  $freq(a) = ?$  or  $1$ . Revising our earlier definition, we call an XML data tree  $T'$  *compatible* with  $G$  if there is a homomorphism  $\phi : V_{T'} \rightarrow V_G$  between  $T'$  and  $G$  such that for every vertex  $v'$  of  $T'$  and every arc  $a = (\phi(v'), w)$  of  $G$  the number of arcs  $a' = (v', w')$  mapped to  $a$  is at most 1 if  $freq(a) = ?$ , at least 1 if  $freq(a) = +$ , and exactly 1 if  $freq(a) = 1$ .

Figure 6 shows an XML schema graph with frequencies assigned to its arcs. At the moment, every course may use an arbitrary number of textbooks. When changing the corresponding frequency from  $*$  to  $+$  or  $1$ , usage of textbooks becomes mandatory. Changing it to  $?$  or  $1$  restricts the usage to a single textbook.

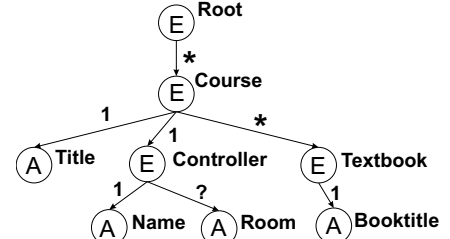


Fig. 6. A XML schema graph with frequencies.

For convenience, we denote by  $G_{\leq 1}$  the subgraph of  $G$  containing all arcs with frequency  $?$  or  $1$ , and by  $G_{\geq 1}$  the subgraph of  $G$  containing all arcs with frequency  $+$  or  $1$ .

The reflexivity axiom, the subtree rule, the supertree rule, the union rule and the generalized union rule from the previous section are still sound in the presence of frequencies. The same holds for the root axiom when redefining  $R$  to be the union of all those  $r_T$ -walks of  $T$  which lie in  $T_{\leq 1}$  and of all those which do not share a non- $T_{\leq 1}$ -arc with any other  $r_T$ -walk of  $T$  and whose single leaf is in  $V_T^E$ . Moreover, the transitivity rule is still not sound in general, but we may apply a *restricted transitivity rule*

$$\frac{X \rightarrow Y, Y \rightarrow Z}{X \rightarrow Z} \text{ } Y \text{ is } X, Z\text{-compliant}, \quad (8)$$

that is,  $Y$  is a subgraph of  $X \cup T_{\geq 1} \cup W$  for every  $r_T$ -walk  $W$  in  $Z$ . Using the restricted transitivity rule we may again obtain a sound and complete system of inference rules for XFDs in Definition 1 and in the presence of frequencies, cf. [14].

## 8. Identifiers

In addition to ordinary attributes, XML allows the specification of identifiers. These are attributes which may be used to equip each element possessing such an attribute with a unique identity. As above, this uniqueness may be reflected by an extended version of the root axiom.

## 9. More functional dependencies

In Definition 1 we introduced functional dependencies  $X \rightarrow Y$  where  $X$  and  $Y$  are  $r_T$ -subgraphs, that is, they share their root with the underlying XML schema graph. These are essentially the functional dependencies which have been suggested in [4].

In the XML documents modelled by the XML schema tree  $T$  in Figure 1, we always stored information on the office of individual course controllers in order to give students advice where to find their teacher. Supposing that every teacher has only one office, we have an XFD  $X \rightarrow Y$  where  $X$  (and  $Y$ ) is the  $v_{Root}$ -paths of  $T$  with the single leaf  $v_{Name}$  (and  $v_{Room}$ , respectively). Now consider the slightly modified XML schema graph  $G$  in Figure 3 and its unfolded version  $T_G$  in Figure 4. Using XFDs provided by Definition 1, we are still able to express that the name of a course controller determines his office as well as that the name of a co-teacher determines his office. Nevertheless, it is well possible that in an XML document the name of a teacher appears together with two different rooms (one in elements of type Controller and another one in elements of type Co-Teacher). As we will see immediately a slight modification of Definition 1 allows us to specify XFDs which may be used to exclude this inconsistency.

Given an XML graph  $G$  and some vertex  $v$  of  $G$ , let  $G(v)$  denote the maximal  $v$ -subgraph of  $G$ , that is, the union of all  $v$ -subgraphs of  $G$ . We call  $G(v)$  the *total  $v$ -subgraph* of  $G$ .

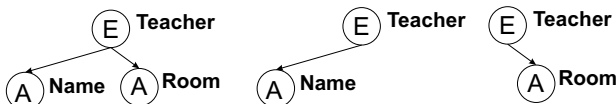


Fig. 7. The total  $v_{Teacher}$ -subgraph and two  $v_{Teacher}$ -subgraphs  $X$  and  $Y$ .

**Definition 2** Let  $G$  be an XML schema graph and consider a total  $v$ -subgraph  $G(v)$  of  $G$ . A functional dependency (or XFD, for short) may be defined as an expression  $v : X \rightarrow Y$  where  $X$  and  $Y$  are  $v$ -subgraphs of  $G(v)$ . Let  $T'$  be an XML data tree compatible with  $G$  and let  $\phi : V_{T'} \rightarrow V_G$  be the unique homomorphism between  $T'$  and  $G$ . Then  $T'$  satisfies the XFD under discussion if for any two almost-copies  $S'_1$  and  $S'_2$  of  $G(v)$  the projections  $S'_1|_Y$  and  $S'_2|_Y$  are equivalent whenever the projections  $S'_1|_X$  and  $S'_2|_X$  are equivalent and copies of  $X$ .

In order to express that the name of a teacher determines his office we may now formulate an XFD  $v_{Teacher} : X \rightarrow Y$  with respect to the XML schema graph  $G$  in Figure 3, where  $X$  and  $Y$  are just the two  $v_{Teacher}$ -subgraphs of  $G$  in Figure 7.

## 10. Even more functional dependencies

Reconsider the XFD  $X \rightarrow Y$  defined with respect to the XML schema graph in Figure 1, where  $X$  and  $Y$  are just the  $v_{Root}$ -subgraphs in Figure 5. The idea was to express that whenever two courses run under the same title and are delivered by the same paper controller they should use the same textbook. When introducing frequencies we did not claim courses to use no more than a single textbook. Figure 9 below shows an XML data tree with courses using several textbooks. The XFD under discussion states, however, that the title of a course together with the name of the course controller determine the booktitle of the textbook to be used. That is, if a single course uses several textbooks, they must all have the same booktitle - which is surely not our intention.

Rather we should use an XFD stating that the title of a course together with the name of the course controller determine the set of booktitles of the textbooks to be used. This, however, is impossible when applying Definition 1. In fact, the XML data tree in Figure 9 below violates the XFD  $X \rightarrow Y$  under discussion. Again a slight modification of Definition 2 provides us a notion of XFDs which is able to cope with the problem mentioned above.

Let  $G'$  and  $G$  be XML graphs together with a homomorphism  $\phi : V_{G'} \rightarrow V_G$  between them.  $\phi$  induces a mapping of the total  $v'$ -subgraphs of  $G'$  to the total  $v$ -subgraphs of  $G$ : given a total  $v'$ -subgraph  $G'(v')$  of  $G'$ ,  $\phi(G'(v'))$  denotes the total  $\phi(v')$ -subgraph  $G(\phi(v'))$  of  $G$ . It should be emphasized that the vertices and arcs of  $\phi(G'(v'))$  are just images of the vertices and arcs of  $G'(v')$ , respectively. Obviously, the number of pre-images of a total  $v$ -subgraph of  $G$  coincides with the number of pre-images of the vertex  $v$ .

**Definition 3** Given an XML schema graph  $G$  and a total  $v$ -subgraph  $G(v)$  of  $G$ , we again define a functional dependency (or XFD, for short) as an expression  $v : X \rightarrow Y$  where  $X$  and  $Y$  are  $v$ -subgraphs of  $G(v)$ . Let  $T'$  be an XML data tree compatible with  $G$  and let  $\phi : V_{T'} \rightarrow V_G$  be the unique homomorphism between  $T'$  and  $G$ . Then  $T'$  satisfies the XFD under discussion if for any two pre-images  $S'_1$  and  $S'_2$  of  $G(v)$  the projections  $S'_1|_Y$  and  $S'_2|_Y$  are equivalent whenever the projections  $S'_1|_X$  and  $S'_2|_X$  are equivalent.

In the example above, we may use Definition 3 to specify XFD  $v_{Course} : X \rightarrow Y$  where  $X$  and  $Y$  are the  $v_{Course}$ -subgraphs in Figure 1 in order to express our intention. It is easy to check that the XML data tree in Figure 9 below satisfies this new functional dependency. Further, it should be emphasized that this dependency is formulated with respect to the vertex  $v_{Course}$  in the XML schema tree  $T$  in Figure 2, not with respect to the root  $v_{Root}$  of  $T$ . This is essential as we are not interested in expressing that the set of titles of all the courses together with the names of the corresponding paper controllers determines the set of all booktitles. (This later XFD is trivially satisfied.) For further details on XFDs in Definition 3, we refer to [13].

## 11. Conclusion

In the present paper we summarized a recent approach towards functional dependencies for XML (called XFDs) based on homomorphisms between XML data trees and XML schema graphs. Our approach allows us to capture functional dependencies as already studied in literature [4, 16] as well as further classes of XFDs. Within this paper we gave examples of XFDs which enable us to describe some essentially different dependencies in XML documents. However, it is fairly easy to modify our definitions to obtain even more examples. This variety of XFDs should not be seen as a bulk of controversial concepts, but as concepts which complement each other. Due to the different expressiveness, all the functional dependencies under discussion could be useful when designing and managing persistent XML documents and should be considered in future work when defining normal forms to describe well-designed XML documents.

## References

1. S. Abiteboul, P. Buneman, and D. Suciu. *Data on the web: From relations to semistructured data and XML*. Morgan Kaufmann, 1999.
2. S. Abiteboul, R. Hull, and V. Vianu. *Foundations of databases*. Addison-Wesley, 1995.
3. M. Arenas, W. Fan, and L. Libkin. What's hard about XML schema constraints. In *DEXA 2002*, pages 269–278, 2002.
4. M. Arenas and L. Libkin. A normal form for XML documents. In *PODS 2002*, pages 85–96, 2002.
5. W. W. Armstrong. Dependency structures of database relationship. *Inform. Process.*, 74:580–583, 1974.
6. W. W. Armstrong, Y. Nakamura, and P. Rudnicki. Armstrong's axioms. *J. Formalized Math.*, 14, 2002.
7. T. Bray, J. Paoli, and C. Sperberg-McQueen, editors. *Extensible Markup Language (XML) 1.0*, W3C Recommendation, 2000.
8. P. Buneman, S. Davidson, W. Fan, and C. Hara. Reasoning about keys for XML. In *International Workshop on Database Programming Languages 2001*. Springer, 2001.
9. P. Buneman, S. Davidson, W. Fan, C. Hara, and W. Tan. Keys for XML. *Computer Networks*, 39:473–487, 2002.
10. B. Courcelle. Fundamental properties of infinite trees. *Theoret. Comput. Sci.*, 25:95–169, 1983.
11. W. Fan and L. Libkin. On XML constraints in the presence of DTDs. *J. ACM*, 49:23–34, 2002.
12. W. Fan and J. Simeon. Integrity constraints for XML. In *PODS 2000*. ACM, 2000.
13. S. Hartmann and S. Link. More functional dependencies for XML. In *ADBIS 2003*. Springer, 2003.
14. S. Hartmann and S. Link. On functional dependencies in XML. submitted for publication, 2003.
15. D. Jungnickel. *Graphs, networks and algorithms*. Springer, 1999.
16. M. Lee, T. Ling, and W. Low. Designing functional dependencies for XML. In *EDBT 2002*, pages 124–141. Springer, 2002.
17. M. Levene and G. Loizou. *A guided tour of relational databases and beyond*. Springer, 1999.
18. B. Thalheim. *Entity-relationship modeling*. Springer, Berlin, 2000.
19. H. Thompson, D. Beech, M. Maloney, and N. Mendelsohn, editors. *XML Schema Part 1: Structures*, W3C Recommendation, 2001.
20. V. Vianu. A web odyssey: From Codd to XML. In *Principles of Database Systems*, 2001.
21. G. E. Weddell. Reasoning about functional dependencies generalized for semantic data models. *ACM Trans. Database Systems*, 17:32–64, 1992.
22. R. K. Wong and W. Shui. Utilizing multiple bioinformatics information sources: An XML database approach. In *BIBE 2001*, pages 73–80. Springer, 2001.

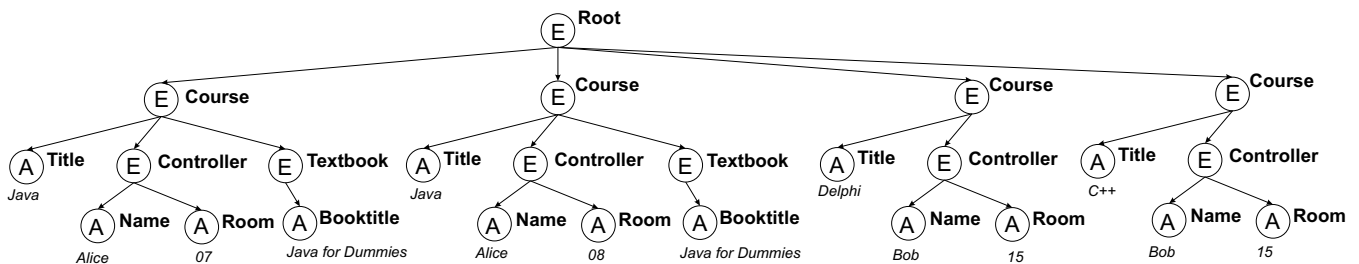


Fig. 8. An XML data tree compatible with the XML schema tree in Figure 1.

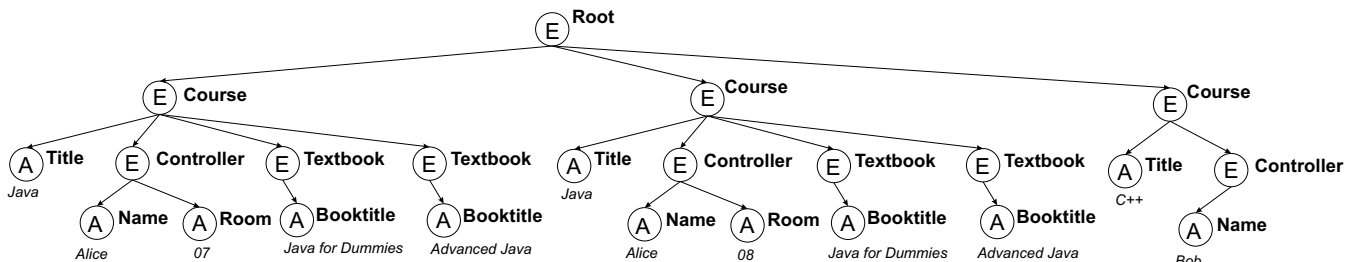


Fig. 9. Another XML data tree compatible with the XML schema tree in Figure 1.