

Solving the Implication Problem for XML Functional Dependencies with Properties

Sven Hartmann¹, Sebastian Link², and Thu Trinh¹

¹ Clausthal University of Technology, Germany

² Victoria University of Wellington, New Zealand

Abstract. Due to the complex nature of XML, finding classes of integrity constraints for XML data that are both expressive and practical is an important but challenging task. In this paper, we study a class of XML functional dependencies (called pXFDs) defined on the basis of tree homomorphism. We establish a semantic equivalence between the implications problems for pXFDs and for propositional Horn clauses, which guarantees linear time decidability of pXFD implication. Hence, pXFDs cannot only be used to capture relevant data semantics, but also be reasoned about efficiently.

1 Introduction

Functional dependencies were first introduced by Codd together with the relational data model [5]. Since then important applications of functional dependencies have been encountered, e.g., in database design, data management, data security, data mining, and data cleansing. Functional dependencies are a valuable aid for capturing the semantics of data, e.g., to express business rules that hold in the the fragment of reality represented by the database. Once specified at design time of a database they can be exploited during run time, e.g., for enforcing data integrity, avoiding update anomalies, and rewriting queries to optimise response times. Most importantly, functional dependencies are the basis for schema normalisation that aims at generating “well-designed” databases.

More recently, the eXtensible Markup Language (XML) has emerged as a well-accepted and widely used data model for heterogenous or complex data in many application domains, including e-science, business integration and service computing. Today, all major DBMS support the storage and processing of XML data. With the increasing amounts of persistent XML data, there is an acute need for developing concepts, algorithms and techniques for efficiently organising and handling XML data. As XML was originally conceived as the W3C standard for exchanging data over the web, it provides only limited capabilities for specifying the semantics of data. Consequently, the study of functional dependencies and other integrity constraints has been identified as an important yet challenging topic of XML research [8,14,17,21].

For the relational data model there is a single ubiquitous notion of functional dependency that can be found in most database textbooks: a *functional dependency (FD)* $X \rightarrow Y$ states that whenever two tuples in a database table agree

on each attribute in X then they must also agree on each attribute in Y . In contrast, the complex nature of XML has sparked a multitude of proposals for *XML functional dependency (XFD)*, including [1,9,18,20]. The varying proposals deviate in their expressiveness, but are all justified by natural occurrence in XML data. Roughly speaking, there are several reasonable options how to generalise the notions of “tuple”, “attribute”, and “agreement” to XML.

The successful application of functional dependencies in relational databases has been based on a thorough investigation of their logical and computational characteristics. As for relational databases, being able to reason about a class of XFDs efficiently will be critical in our ability to apply XFDs for tasks such as integrity enforcement, query optimisation, and schema normalisation. So far, only very few classes of XFDs are known to have tractable logical implication and, even then, only under certain conditions. Thus the search for practical formalisms of XML functional dependency that can be reasoned about efficiently remains a challenge.

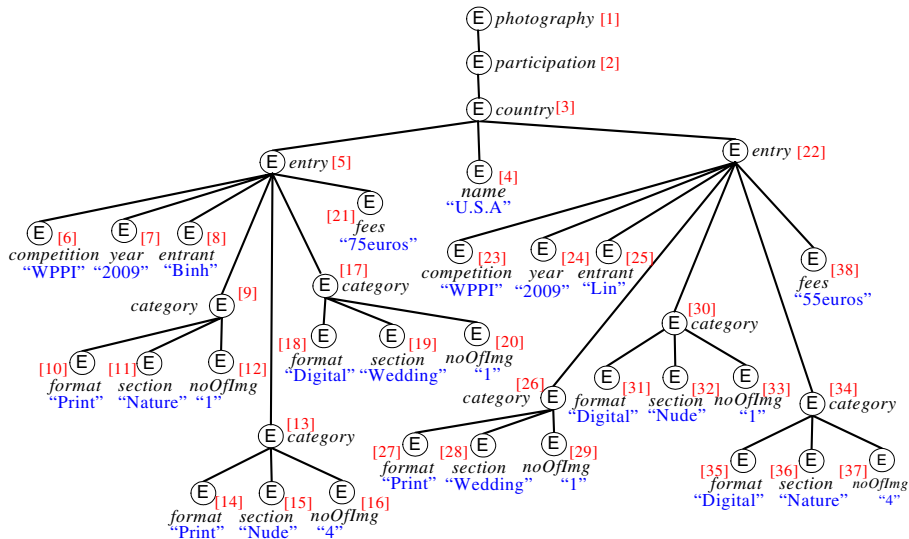


Fig. 1. An XML data tree T'_{photo} with photographic entries

Example 1.1. In this paper, we are concerned with a class of XFDs that extend a proposal in [9]. For motivation, consider an XML document with data on images entered into a photography competition¹, depicted as a data tree in Fig. 1. It is regular practice for the competition organisers to fix the pricing plan every year based on the number of images entered per section per format.

Suppose, for example, the entry fees charged for the WPPI¹ competition 2009 were as follows:

format:	Digital	Print
1 image in 1 section	10 euros	15 euros
4 images in 1 section	30 euros	50 euros

¹ cf. Wedding & Portrait Photographers International, <http://www.wppionline.com/>

In T'_{photo} only the total fees paid by each entrant are recorded, as the pricing schedule is stored in a separate place. It is easy to see that we have the following integrity constraint:

Constraint 1. *For every competition and year, the collection of information about the number of images entered into each section in each format determines the total fees of an entry.*

Using a path-based XFD as proposed, e.g., in [1,18,22], the closest that we come to expressing the constraint above is as follows:

$$\left. \begin{array}{l} \text{photography/participation/country/entry/competition,} \\ \text{photography/participation/country/entry/year,} \\ \text{photography/participation/country/entry/category/format,} \\ \text{photography/participation/country/entry/category/section,} \\ \text{photography/participation/country/entry/category/noOfImg} \end{array} \right\} \\ \rightarrow \{ \text{photography/participation/country/entry/fees} \}$$

But this is not what we want to express: In the data tree T'_{photo} , there are two entrants for WPPI 2009 having the same collection of formats, the same collection of sections, and the same collection of numbers of images, but the fees charged to them are different. This means that the path-based XFD above is violated in T'_{photo} . On the other hand, however, the data tree T'_{photo} satisfies Constraint 1 as it complies with the given pricing schedule.

Constraint 1 helps to pinpoint two limitations of most XFD proposals in the literature: agreement is decided by *i*) comparing singular data items (rather than collections), and *ii*) considering paths independently from one another (rather than jointly as a tree structure). Both issues motivated the XFD proposal in [9] which is capable of expressing integrity constraints like Constraint 1: it allows subtrees of the schema tree to occur as “attributes” in the left and right hand side of an XFD. Here, we extend this idea further by considering an even wider notion of “attributes”. For motivation, take the following integrity constraint:

Constraint 2. *For every entry, the information about the format and section determines the category.*

In other words, the constraint states that no entry has multiple categories with the same format and section. Such a functional dependency stipulates a uniqueness constraint that allows one to identify specific nodes within the data tree. This is particularly useful when looking for a syntactic characterisation for the absence of data redundancy - an important indicator of a well-designed schema.

Contribution. We study a class of XFDs that extend the proposal in [9]. We prove the associated implication problem to be equivalent to that of propositional Horn clauses, thus generalising an important result of Fagin for relational FDs. This gives rise to a linear-time algorithm for deciding implication for these XFDs. Thus we have found a new class of XFDs that are both expressive and practical. Incidentally, our proof of semantic equivalence also shows that finite and unrestricted implication coincides for the class of XFDs under inspection.

Organisation. Section 2 surveys related work, and Section 3 assembles basic terminology. In Section 4 we define the class of XFDs considered in this paper. Section 5 contains our major result on the semantic equivalence of logical implication for these XFDs and propositional Horn clauses. We give an example in Section 6 and conclude in Section 7 with consequences and future work plans.

2 Related Work

The efficiency with which we are able to reason about functional dependencies (i.e., decide implication) plays an important role in our ability to capitalise on their applications. The implication problem for relational FDs has been well-studied: The first axiomatisation for the implication of relational FDs was given by Armstrong [2]. Thus FD implication can be tackled by examining the enumeration of possible derivations. Several algorithmic approaches have emerged, often based on the Armstrong axioms. One of the earliest is due to Beeri and Bernstein [3]. The implication problem for relational FDs turned out to be solvable in linear time. This complexity result can also be concluded from the seminal work of Fagin [7] who related the implication of relational FDs to propositional logic. In fact, he proved that the implication problems for relational FDs and for propositional Horn clauses coincide. For propositional Horn clauses, however, the implication problem is equivalent to the well-studied satisfiability problem which can be solved in linear time, e.g., through unit propagation [4,6].

In the case of XML, the implication problem has been studied for a few selected classes of XFDs. Arenas and Libkin [1] showed that for their “tree-tuple” XFDs the implication problem can be solved in polynomial time in the presence of two restricted kinds of DTDs. In particular, for simple DTDs which corresponds to our schema trees, they obtained a quadratic time algorithm. In the presence of two other kinds of DTDs they prove the implication problem for “tree-tuple” XFDs to be coNP-complete. In general, however, they notice that the implication of “tree-tuple” XFDs is not finitely axiomatisable due to a non-trivial interaction with DTDs. Hartmann and Trinh [11,15] gave a finite axiomatisation for the implication of “tree-tuple” XFDs in the presence of schema trees, while Kot and White [12] found a finite axiomatisation for the implication of “tree-tuple” XFDs in the absence of a DTD and in the presence of some special kinds of DTDs. An axiomatisation for “pre-image” XFDs proposed in [9] was sketched in [10]. Vincent et al. [18] gave a sound set of inference rules for the implication of their class of “closest node” XFDs which, however, is only complete for the special case where the left hand sides contain no more than a single path. The relationship of “closest node” XFDs and “tree-tuple” XFDs has been discussed in [19]: both XFD proposals only coincide under very strong assumptions that severely detract from the syntactic flexibility of XML.

The XFDs studied in this paper complement the expressive power of existing classes of path-based XFDs, in particular “tree-tuple” and “closest node” XFDs. There are meaningful functional dependencies that can be expressed as the former but not the latter, and vice versa.

3 Preliminaries

An XML tree is a rooted tree T with finite node set V_T , arc set A_T , root r_T , and mappings $name : V_T \rightarrow Names$ and $kind : V_T \rightarrow \{E, A\}$. In an XML tree, the symbols E and A indicate elements and attributes, with attributes only appearing as leaf nodes. An XML data tree is an XML tree T' with a mapping $valuation : L_{T'} \rightarrow String$ assigning string values to leaves. An XML schema tree is an XML tree T with frequencies $?, 1, *, +$ assigned to its arcs, where *i*) arcs to attribute nodes may only have frequency $?$ or 1 and, *ii*) no two siblings have the same name and kind.

We use XML schema trees as structural summaries for collections of XML data trees. The notion of *compatibility* conveys that a particular XML data tree conforms to the structural summary given by an XML schema tree T at hand. We say that an XML data tree T' is T -compatible whenever there is a homomorphism $\phi : V_{T'} \rightarrow V_T$ (i.e., root-preserving, name-preserving, kind-preserving and arc-preserving mapping) such that for every vertex v' of T' and every arc $a = (\phi(v'), w)$ of T , the number of arcs $a' = (v', w')$ mapped to a is at most one if a has frequency label $?$, exactly one if a has frequency label 1 , at least one if a has frequency label $+$, and arbitrarily many if a has frequency label $*$.

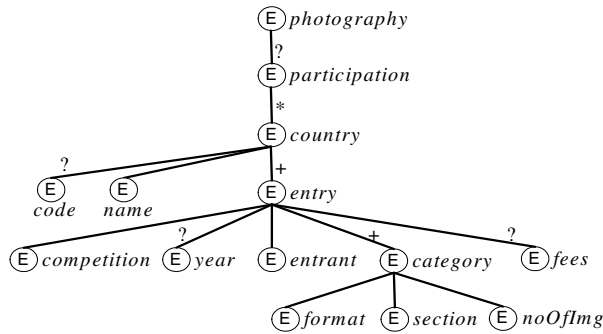


Fig. 2. The data tree T'_{photo} is compatible to the XML schema tree T_{photo} shown here

For every node v in an XML tree there is a unique path from the root to v . In an XML schema tree T we call a path *simple* iff it contains no arcs with frequency other than $?$ and 1 . Furthermore, we call a node v in T *simple* iff the path from the root to v is simple. Given a node v , a node n which has a (possibly empty) path to v is called a v -*ancestor*. We can also say v is a *descendant* of n or n is an *ancestor* of v . By $\check{A}_T(v)$ we denote the set of all v -ancestors of T . Furthermore, n is a *simple ancestor* of v (or equivalently v is a *simple descendant* of n) iff the path connecting n with v is simple. Clearly, each node is also its own (simple) ancestor and (simple) descendant, and each simple node is a simple descendant of the root node. We use v_{lbl} to refer to a node with name lbl .

Let T be an XML tree, and $L_T \subseteq V_T$ be a given set of leaves of T . A *walk* of T is a path from the root of T to a member of L_T and every walk containing v is

called a v -walk. A *subgraph* W of T is a (possibly empty) set of walks of T and, a subgraph of T is a v -subgraph iff each of its walks contains v . By $\check{S}_T(v)$ we denote the set of all v -subgraphs of T . It is easy to see that $\check{S}_T(v)$ contains the empty set and is closed under the union, intersection and difference operators. Clearly a walk or subgraph of T is again an XML tree.

For convenience, we made an effort to use only XML schema trees whose leaves have mutually distinct names as examples in this paper. This saves space as it allows us to refer to a walk by the name of its leaf, and correspondingly we refer to a subgraph by a set of leaf names. Moreover, for a subgraph \mathcal{X} consisting of a single walk B we tend to write B instead of $\{B\}$. Note that we use these abbreviations due to space limitations and do not exclude other cases.

The *total v -subgraph*, denoted by $T(v)$, is the set of all v -walks of XML tree T . The homomorphism ϕ between a T -compatible data tree T' and schema tree T induces a mapping of the total subgraphs of T' to the total subgraphs of T . For a fixed node v of T , a *pre-image tree of v* in T' is just a total w -subgraph with $\phi(w) = v$. Suppose every node in a data tree has a unique node id, then the node id for a node w in the data tree also identifies the total w -subgraph in the data tree. By $P_{T'}(v)$ we denote the set of all pre-image trees of v in T' .

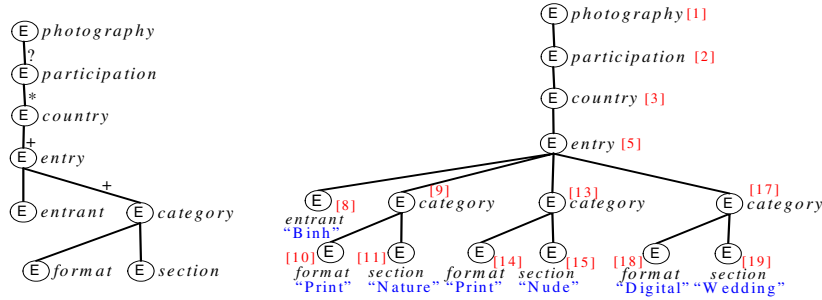


Fig. 3. A subgraph $\{\text{entrant}, \text{format}, \text{section}\}$ of T_{photo} and the projection of the pre-image tree i_2 of v_{entry} to this subgraph

4 XML Functional Dependencies with Properties

Next we formally define the class of XFDs investigated in this paper. Throughout let T be an XML schema tree, T' a T -compatible data tree, and v a node of T . As discussed we need to say how we want to generalise the notions of “tuple”, “attribute”, and “agreement” from the definition of relational FDs. In the subsequent definition, we will use the v -pre-image trees as “tuples”, while the v -ancestors and v -subgraphs will be used as “attributes”. For short, we will collectively call the members of $\check{A}_T(v) \cup \check{S}_T(v)$ the v -properties of T .

“Agreement” will be defined on the basis of tree homomorphism. Given two XML trees T_1 and T_2 , we say that they are *isomorphic* or *copies* of one another if there is a homomorphism $\phi : V_{T_1} \rightarrow V_{T_2}$ which is bijective and ϕ^{-1} is a homomorphism. In particular, we call such a mapping ϕ a *(tree) isomorphism*. A subgraph U of T_1 is a *subcopy* of T_2 if U is isomorphic to some subgraph of T_2 .

To explain when two pre-image trees “agree” on a v -property we need to state what the projection of a pre-image tree to a v -property is. Intuitively projecting to a v -property yields either nodes or XML trees depending on whether we consider a v -ancestor or v -subgraph, and thus we have two cases:

Definition 4.1 (projection). *For a v -property X of T , the projection of T' to X , denoted by $T'|_X$, is:*

- the set of all pre-images of X in T' , if X is a v -ancestor of T , or
- the union of all subcopies of X in T' , if X is a v -subgraph of T .

In the literature, two notions of agreement are popular when comparing fragments of XML data: node-equality and value-equality. We use node-equality when comparing the projection to v -ancestors, and value-equality when comparing projection to v -subgraphs.

Definition 4.2 (property-equality, \doteq). *Property-equality holds as follows:*

- Two sets p, q of nodes in an XML data tree T' are property-equal iff $p = q$.
- Two XML data trees p, q are property-equal iff there exists a valuation-preserving isomorphism $\phi : V_p \rightarrow V_q$ between p and q .

In words, we say “pre-image trees p_1, p_2 agree on X ” to mean $p_1|_X \doteq p_2|_X$. Likewise, we say “pre-image trees p_1, p_2 differ on X ” to mean $p_1|_X \not\doteq p_2|_X$.

Definition 4.3 (pXFD). *An XML functional dependency with properties (pXFD) over T is an expression $v : \mathcal{X} \rightarrow \mathcal{Y}$ where \mathcal{X}, \mathcal{Y} are sets of v -properties of T . Herein, v is called the target, \mathcal{X} the LHS, and \mathcal{Y} the RHS.*

T' satisfies the pXFD $v : \mathcal{X} \rightarrow \mathcal{Y}$, written as $\models_{T'} v : \mathcal{X} \rightarrow \mathcal{Y}$, iff for any two pre-image trees $p_1, p_2 \in P_{T'}(v)$ we have $p_1|_X \doteq p_2|_X$ for all $X \in \mathcal{X}$ imply $p_1|_Y \doteq p_2|_Y$ for all $Y \in \mathcal{Y}$. We also say that the pXFD holds in T' .

Note that we *never* omit outer set parentheses for *sets* of v -properties. In particular, a (possibly empty) v -property is *always* enclosed in set parentheses. \emptyset denotes the empty v -subgraph, while $\{\}$ denotes the empty set of v -properties.

Example 4.1. The data tree T'_{photo} in Fig. 1 satisfies the pXFDs

$$v_{entry} : \{\text{competition, year, \{format, section, noOfImg\}}\} \rightarrow \{\text{fees}\} \quad (1)$$

$$v_{category} : \{v_{entry}, \{\text{format, section}\}\} \rightarrow \{v_{category}\} \quad (2)$$

which capture Constraints 1 and 2, respectively. But T'_{photo} violates the pXFDs

$$v_{entry} : \{\text{competition, year, format, section, noOfImg}\} \rightarrow \{\text{fees}\}$$

$$v_{entry} : \{\{\text{format, section}\}\} \rightarrow \{v_{entry}\}.$$

We like to emphasise that the addition of v -ancestors as “attributes” to XML functional dependencies is quite a powerful extension. A v -ancestor in the left hand side of a pXFD (as for example v_{entry} in (2)) may be regarded as a *context*

that restricts the validity of a functional dependency to parts of the entire XML data tree (here the pre-image trees of v_{entry}).

A v -ancestor on the right hand side of a pXFD may be used to capture a categorisation or grouping of nodes. For example, the pXFD

$$v_{entry} : \{\mathbf{competition}\} \rightarrow \{v_{country}\}$$

expresses that all entries for the same competition must be grouped under the same country. Probably most interesting is the inclusion of the target v into the right hand side of a pXFD to capture the uniqueness of nodes. For example, $v_{category}$ in the right hand side of the pXFD (2) just reflects the uniqueness constraint stipulated by Constraint 2 in the introduction.

Syntactically, one can specify a large number of pXFDs since the number of subsets of v -properties is exponential in the number of v -properties. In practice, however, we can restrict ourselves to a smaller subclass of pXFDs without losing expressiveness. We call the pXFDs in this subclass *canonical*. Each pXFD can be rewritten as a canonical pXFD by translating non-essential v -properties into *essential* ones. We give details in the appendix.

Our paper is motivated by the implication problem for XML functional dependencies. A set of pXFDs Σ *implies* a pXFD σ , denoted by $\Sigma \models \sigma$, iff every T -compatible data tree which satisfies all pXFDs in Σ also satisfies σ . The *pXFD implication problem* is to decide, given any set $\Sigma \cup \{\sigma\}$ of pXFDs, whether $\Sigma \models \sigma$.

5 The Semantic Equivalence Theorem

Our objective is to establish a semantic equivalence between the implication of pXFDs and propositional Horn clauses that generalises the seminal result of Fagin for relational FDs to XML. For that, we need a *Horn encoding* that uses propositional horn clauses for encoding the given pXFDs but also the inherent structural properties associated with the underlying XML tree. We assume some familiarity with propositional logic, cf. [13]. A *Horn clause* over some given set of literals \mathcal{V} is a *clause* (i.e., a disjunction of literals) with at most one *positive* (i.e., non-negated) literal.

Let $\varphi : \mathbf{E}_T^{\mathbb{S}}(v) \cup \mathbf{E}_T^{\mathbb{A}}(v) \rightarrow \mathcal{V}$ be a mapping that assigns propositional variables to the essential v -properties of T . If σ is a canonical pXFD $v : \{X_1, \dots, X_j\} \rightarrow \{Y_1, \dots, Y_k\}$ then, let

$$H_\sigma = \{\varphi(X_1) \wedge \dots \wedge \varphi(X_j) \Rightarrow \varphi(Y_1), \dots, \varphi(X_1) \wedge \dots \wedge \varphi(X_j) \Rightarrow \varphi(Y_k)\}$$

be the Horn encoding of σ . For a set Σ of pXFDs, let H_Σ be the union of the sets H_σ for all $\sigma \in \Sigma$. Furthermore, we capture information about inherent inter-relationships among essential v -properties by the *base translation*:

$$\begin{aligned} H_T = & \{\varphi(W) \Rightarrow \varphi(Z) \mid W, Z \in \mathbf{E}_T^{\mathbb{S}}(v) \text{ and } W \text{ covers } Z \text{ and } Z \neq \emptyset\} \\ & \cup \{\varphi(n) \Rightarrow \varphi(m) \mid n, m \in \mathbf{E}_T^{\mathbb{A}}(v) \text{ and } m \text{ is an immediate essential} \\ & \quad \text{ancestor of } n \text{ and } n \neq r_T\} \\ & \cup \{\varphi(n) \Rightarrow \varphi(U) \mid \{n\} = \vartheta(\{v\}) \text{ and } U \text{ is a } v\text{-unit}\} \\ & \cup \{\varphi(\emptyset), \varphi(r_T)\} \end{aligned}$$

where a v -subgraph W is said to *cover* a v -subgraph Z iff W is the union of Z and just one additional v -walk of T , and an essential v -ancestor m is an *immediate essential ancestor* of another essential v -ancestor n iff n is the next essential v -ancestor which can be reached from m .

An example for the Horn encoding is given in Section 6. To decrease the size of the base translation, we made use of the transitive nature of logical implication to consider only *covering* v -subgraphs and *immediate essential ancestor/descendant* essential v -ancestors. It is, of course, possible to use the more general notion of subgraph containment and ancestor/descendant relationship but this is likely to result in a considerably larger set of Horn clauses for the base translation.

Using the Horn encoding above we state our main result in this paper:

Theorem 5.1 (Semantic Equivalence Theorem for pXFDs). *The following statements are equivalent:*

1. Σ implies σ ,
2. Σ implies σ in the world of two- v -pre-image data trees,
3. $H_\Sigma \cup H_T$ logically implies H_σ .

Σ is said to *imply* σ in the world of two- v -pre-image data trees iff every T -compatible data tree containing precisely two pre-image trees of v that satisfies all pXFDs in Σ also satisfies σ . To prove (1. \Leftrightarrow 2.) we show that every T -compatible data tree that witnesses $\Sigma \models \sigma$ to be *false* contains a two- v -pre-image data tree that already witnesses the same fact.

The proof of (2. \Leftrightarrow 3.) relies on a special connection between boolean assignment for propositional variables and the agreement of two pre-image trees of v . With this connection, it is possible to determine a boolean (truth) assignment witnessing a logical implication of Horn clauses from a two- v -pre-image T -compatible XML data tree witnessing the corresponding canonical pXFD implication, and vice versa. We use proof by the contrapositive. One direction follows easily from the characteristics of such representative boolean assignments, while the other direction additionally relies on the ability to construct a two- v -pre-image data tree from some input boolean assignment \mathbb{B} such that the the resulting data tree has \mathbb{B} as a representative boolean assignment.

For the proof of (2. \Leftrightarrow 3.) we need to relate boolean assignments to two- v -pre-image data trees in such a way that pXFDs are satisfied in T' precisely when their corresponding Horn clauses evaluate to *true* under the boolean assignment:

Definition 5.1 (representative boolean assignment). *Let T be an XML schema tree and $\varphi : \mathbf{E}_T^{\hat{S}}(v) \cup \mathbf{E}_T^{\hat{A}}(v) \rightarrow \mathcal{V}$ be a mapping assigning propositional variables to the essential v -properties of T . Further, let \mathbb{B} be a boolean assignment of all propositional variables in \mathcal{V} . The boolean assignment \mathbb{B} is said to be representative of a given T -compatible two- v -pre-image data tree T' where $P_{T'}(v) = \{p_1, p_2\}$ iff the equivalence $\mathbb{B}(\varphi(W)) = \text{true}$ iff $p_1|_W \doteq p_2|_W$ holds for every essential v -property $W \in \mathbf{E}_T^{\hat{S}}(v) \cup \mathbf{E}_T^{\hat{A}}(v)$.*

Two important characteristics of boolean assignments \mathbb{B} representing two- v -pre-image data trees are crucial here. The first lemma is analogous to the Semantic Lemma of Fagin [7] and relates the satisfaction of pXFDs in T' to the truth value of the corresponding Horn clauses under \mathbb{B} . The second lemma affirms that every Horn clause belonging to base translation H_T evaluates to *true* under \mathbb{B} . This attests to the fact that the base translation only encodes inherent inter-relationships among essential v -properties with respect to property-equality.

Lemma 5.1 (Semantic Lemma). *Let \mathbb{B} be a boolean assignment which is representative of some T -compatible two- v -pre-image data tree T' . Let σ be a canonical pXFD $v : \mathcal{X} \rightarrow \mathcal{Y}$ over T . Then σ holds in T' iff every Horn clause in the set H_σ evaluates to true under \mathbb{B} .*

Lemma 5.2 (Triviality Lemma). *Let \mathbb{B} be a boolean assignment which is representative of some T -compatible data tree T' . Then every Horn clause in H_T evaluates to true under \mathbb{B} .*

For the proof of (2. \Leftrightarrow 3.) we further need to construct a T -compatible two- v -pre-image data tree having some specified representative boolean assignment. That is, we create a T -compatible two- v -pre-image data tree whose pre-image trees of v agree precisely on some given set of v -properties. From the triviality lemma and our earlier observations about property-equality when identifying essential v -properties, it is clear that the input set of v -properties cannot be arbitrary, rather it must possess certain characteristics:

Definition 5.2 (equality set). *An equality set \mathcal{E}_T is a subset of v -properties such that the following conditions all hold:*

- *The subset of v -ancestors (i.e., $\check{\mathbb{A}}\mathcal{E}_T = \mathcal{E}_T \cap \check{\mathbb{A}}_T(v)$) must contain the root node r_T ; be complete (i.e., if a v -ancestor is in the set then all of its ancestors are also in the set); if a v -ancestor is in the set then all of its simple descendants are also in the set; and not contain the target node v .*
- *The subset of v -subgraphs (i.e., $\check{\mathbb{S}}\mathcal{E}_T = \mathcal{E}_T \cap \check{\mathbb{S}}_T(v)$) must contain the empty subgraph \emptyset ; be complete (i.e., if a v -subgraph is in the set then all v -subgraph contained in it are also in the set); and if two v -reconcilable v -subgraph X, Y are in the set then also v -subgraph $X \cup Y$ is in the set.*

We use a two-phase combinatorial approach for constructing a two- v -pre-image data tree that agrees precisely on the v -properties in a given equality set:

1. Firstly, we construct two distinct T -compatible pre-image trees of v which agree precisely on v -subgraphs in the given input equality set.
2. Secondly, we merge the constructed pre-image trees to form a T -compatible data tree in which the pre-image trees share precisely the v -ancestors in the given input equality set.

There is no conflict between the two phases since the former deals exclusively with descendants of v and the latter only with nodes not descended from v . An outline of the construction is given in the appendix. For a complete proof of Theorem 5.1 we refer the interested reader to [16].

6 An Example for Applying the Semantic Equivalence

The following example demonstrates how Theorem 5.1 can be used to decide implication of pXFDs. Recall the XML schema tree T_{photo} in Fig. 2 and consider its node v_{entry} . The essential v_{entry} -properties are $v_{participation}$, $v_{country}$, v_{entry} , **competition**, **year**, **entrant**, and all subgraphs contained in **{format,section,noOfImg}**.

We study an instance of the pXFD implication problem. Let σ be the pXFD $v_{entry} : \{\mathbf{competition}, \mathbf{year}, \mathbf{entrant}\} \rightarrow \{\mathbf{fees}\}$, and let Σ consist of the pXFDs $v_{entry} : \{\mathbf{competition}, \{\mathbf{format}, \mathbf{section}, \mathbf{noOfImg}\}\} \rightarrow \{\mathbf{fees}\}$, and $v_{entry} : \{\mathbf{competition}, \mathbf{year}, \mathbf{entrant}\} \rightarrow \{v_{entry}\}$.

For a fixed mapping φ from the set of essential v_{entry} -properties to propositional variables, the Horn encoding introduced above yields the following:

$$\begin{aligned}
 H_\sigma &= \{ \varphi(\mathbf{competition}) \wedge \varphi(\mathbf{year}) \wedge \varphi(\mathbf{entrant}) \Rightarrow \varphi(\mathbf{fees}) \} \\
 H_\Sigma &= \left\{ \begin{array}{l} \varphi(\mathbf{competition}) \wedge \varphi(\{\mathbf{format}, \mathbf{section}, \mathbf{noOfImg}\}) \Rightarrow \varphi(\mathbf{fees}), \\ \varphi(\mathbf{competition}) \wedge \varphi(\mathbf{year}) \wedge \varphi(\mathbf{entrant}) \Rightarrow \varphi(v_{entry}) \end{array} \right\} \\
 H_{T_{photo}} &= \left\{ \begin{array}{l} \varphi(\{\mathbf{format}, \mathbf{section}, \mathbf{noOfImg}\}) \Rightarrow \varphi(\{\mathbf{format}, \mathbf{section}\}), \\ \varphi(\{\mathbf{format}, \mathbf{section}, \mathbf{noOfImg}\}) \Rightarrow \varphi(\{\mathbf{format}, \mathbf{noOfImg}\}), \\ \varphi(\{\mathbf{format}, \mathbf{section}, \mathbf{noOfImg}\}) \Rightarrow \varphi(\{\mathbf{section}, \mathbf{noOfImg}\}), \\ \varphi(\{\mathbf{format}, \mathbf{section}\}) \Rightarrow \varphi(\mathbf{format}), \\ \varphi(\{\mathbf{format}, \mathbf{section}\}) \Rightarrow \varphi(\mathbf{section}), \\ \varphi(\{\mathbf{format}, \mathbf{noOfImg}\}) \Rightarrow \varphi(\mathbf{format}), \\ \varphi(\{\mathbf{format}, \mathbf{noOfImg}\}) \Rightarrow \varphi(\mathbf{noOfImg}), \\ \varphi(\{\mathbf{section}, \mathbf{noOfImg}\}) \Rightarrow \varphi(\{\mathbf{section}\}), \\ \varphi(\{\mathbf{section}, \mathbf{noOfImg}\}) \Rightarrow \varphi(\{\mathbf{noOfImg}\}), \\ \varphi(v_{entry}) \Rightarrow \varphi(v_{country}), \\ \varphi(v_{entry}) \Rightarrow \varphi(\mathbf{competition}), \\ \varphi(v_{entry}) \Rightarrow \varphi(\mathbf{year}), \\ \varphi(v_{entry}) \Rightarrow \varphi(\mathbf{entrant}), \\ \varphi(v_{entry}) \Rightarrow \varphi(\{\mathbf{format}, \mathbf{section}, \mathbf{noOfImg}\}), \\ \varphi(v_{entry}) \Rightarrow \varphi(\mathbf{fees}), \\ \varphi(\emptyset), \\ \varphi(v_{participation}) \end{array} \right\}
 \end{aligned}$$

To decide whether σ is implied by Σ , one can check whether there is a boolean assignment which makes all formulas in $H_\Sigma \cup H_{T_{photo}}$ become *true* and the single formula $h \in H_\sigma$ becomes *false*. To make h become *false*, we need to set $\mathbb{B}(\varphi(\mathbf{competition})) = \mathit{true}$, $\mathbb{B}(\varphi(\mathbf{year})) = \mathit{true}$, $\mathbb{B}(\varphi(\mathbf{entrant})) = \mathit{true}$, and $\mathbb{B}(\varphi(\mathbf{fees})) = \mathit{false}$. Then, to make the second formula in H_Σ become *true* we must further set $\mathbb{B}(\varphi(v_{entry})) = \mathit{true}$. Further, to make the formulas in $H_{T_{photo}}$ become *true*, we need to set $\mathbb{B}(\varphi(\{\mathbf{format}, \mathbf{section}, \mathbf{noOfImg}\})) = \mathit{true}$.

However, the resulting assignment \mathbb{B} is invalid as it causes the first formula in H_Σ to become *false*. Hence, we cannot find a boolean assignment as required.

That is, H_σ is a logical consequence of $H_\Sigma \cup H_{T_{photo}}$ and we can correspondingly conclude that Σ implies σ .

7 Conclusion and Future Directions

The Semantic Equivalence Theorem enables us to solve the pXFD implication problem via Horn satisfiability: $\Sigma \models \sigma$ holds unless H_σ is *not* a logical consequence of $H_\Sigma \cup H_T$. This corresponds to the question: For some $h \in H_\sigma$, is there a boolean assignment \mathbb{B} of the propositional variables such that each propositional formula in the set $H_\Sigma \cup H_T \cup \{-h\}$ evaluates to *true*? This is the Horn Satisfiability problem (Horn-SAT) which is decidable in linear time [4,6].

Corollary 7.1. *The problem of whether Σ implies σ can be decided in time linear in the total number of essential v -properties of T .*

Thus, this paper identifies a new class of XFDs for which implication can be decided efficiently. This makes it practical for XML data architects to use our XFDs as integrity constraints in XML design to capture relevant data semantics. This contributes to the general objective of studying the trade-off between expressiveness and tractability of integrity constraints for XML.

The Semantic Equivalence Theorem has some immediate applications: we successfully used it for identifying and proving an axiomatisation for pXFDs, and for decision support in constraint acquisition, cf. [16]. We further record:

Corollary 7.2. *The finite and the unrestricted (i.e., when considering infinite XML data trees, too) implication problem of pXFDs coincide.*

Several areas emerge as possible directions for future work. It would be helpful to establish a similar semantic equivalence for other classes of XFDs, but also for pXFDs in the presence of further classes of DTDs such as #-DTDs or DTDs with a small number of disjunctions. The study of XML functional dependencies is motivated by the large number of potential applications, e.g., in database design, database tuning, or query optimisation as for relational FDs. The investigation of such opportunities will further justify the practicality of the class of pXFDs. We have started investigating XML normal forms based on pXFDs, cf. [16].

References

1. Arenas, M., Libkin, L.: A normal form for XML documents. *ACM ToDS* 29, 195–232 (2004)
2. Armstrong, W.W.: Dependency structures of data base relationships. In: *IFIP Congress*, pp. 580–583 (1974)
3. Beeri, C., Bernstein, P.A.: Computational problems related to the design of normal form relational schemas. *ACM ToDS* 4, 30–59 (1979)
4. Chang, C., Lee, R.: *Symbolic Logic and Mechanical Theorem Proving*. Academic Press, London (1987)
5. Codd, E.F.: Further normalization of the data base relational model. *IBM Research Report RJ909* (1971)
6. Dowling, W.F., Gallier, J.H.: Linear-time algorithms for testing the satisfiability of propositional Horn formulae. *J. Log. Program.* 1, 267–284 (1984)

7. Fagin, R.: Functional dependencies in a relational data base and propositional logic. IBM J. Research Dev. 21, 543–544 (1977)
8. Fan, W.: XML constraints: Specification, analysis, and applications. In: Andersen, K.V., Debenham, J., Wagner, R. (eds.) DEXA 2005. LNCS, vol. 3588, pp. 805–809. Springer, Heidelberg (2005)
9. Hartmann, S., Link, S.: More functional dependencies for XML. In: Kalinichenko, L.A., Manthey, R., Thalheim, B., Wloka, U. (eds.) ADBIS 2003. LNCS, vol. 2798, pp. 355–369. Springer, Heidelberg (2003)
10. Hartmann, S., Link, S., Trinh, T.: Efficient reasoning about XFDs with pre-image semantics. In: Kotagiri, R., Radha Krishna, P., Mohania, M., Nantajeewarawat, E. (eds.) DASFAA 2007. LNCS, vol. 4443, pp. 1070–1074. Springer, Heidelberg (2007)
11. Hartmann, S., Trinh, T.: Axiomatizing functional dependencies for XML with frequencies. In: Dix, J., Hegner, S.J. (eds.) FoIKS 2006. LNCS, vol. 3861, pp. 159–178. Springer, Heidelberg (2006)
12. Kot, L., White, W.M.: Characterization of the interaction of XML functional dependencies with DTDs. In: Schwentick, T., Suciu, D. (eds.) ICDT 2007. LNCS, vol. 4353, pp. 119–133. Springer, Heidelberg (2006)
13. Rautenberg, W.: A Concise Introduction to Mathematical Logic. Springer, Heidelberg (2006)
14. Suciu, D.: On database theory and XML. SIGMOD Rec. 30, 39–45 (2001)
15. Trinh, T.: Axiomatizing functional dependencies for XML with frequencies. Master’s thesis, Massey University (2004)
16. Trinh, T.: XML Functional Dependencies based on Tree Homomorphisms. PhD thesis, Clausthal University of Technology (2009)
17. Vianu, V.: A Web odyssey: from Codd to XML. SIGMOD Rec. 32, 68–77 (2003)
18. Vincent, M.W., Liu, J., Liu, C.: Strong functional dependencies and their application to normal forms in XML. ACM ToDS 29, 445–462 (2004)
19. Vincent, M.W., Liu, J., Mohania, M.K.: On the equivalence between FDs in XML and FDs in relations. Acta Inf. 44, 207–247 (2007)
20. Wang, J., Topor, R.W.: Removing XML data redundancies using functional and equality-generating dependencies. In: ADC 2005, pp. 65–74 (2005)
21. Widom, J.: Data management for XML: Research directions. IEEE Data Eng. Bull. 22, 44–52 (1999)
22. Yu, C., Jagadish, H.V.: XML schema refinement through redundancy detection and normalization. VLDB J. 17, 203–223 (2008)

Appendix

We use the appendix to assemble some more technical details that may provide further insight into the results presented in this paper.

Canonical XFDs. Firstly, we explain when a v -property is essential, and how the proposed translation ϑ into canonical pXFDs works. Let $\mathbf{E}_T^{\check{\Delta}}(v)$ be the set of all v -ancestors whose incoming arc has frequency other than ? or 1. We call the members of $\mathbf{E}_T^{\check{\Delta}}(v)$ *essential v -ancestors*. To check property-equality of a set $\mathcal{X}^{\check{\Delta}}$ of v -ancestors, it is sufficient to consider the *lowest v -ancestor* in this set. Instead of such a v -ancestor, it is equivalent to consider its *highest simple ancestor*, which is still essential. Thus, if $\mathcal{X}^{\check{\Delta}}$ is non-empty then $\vartheta(\mathcal{X}^{\check{\Delta}})$ denotes the singleton set containing the highest simple ancestor of the lowest contained node $\text{lca}(\mathcal{X}^{\check{\Delta}}) \in \mathcal{X}^{\check{\Delta}}$. Otherwise, if $\mathcal{X}^{\check{\Delta}}$ is empty, then $\vartheta(\mathcal{X}^{\check{\Delta}}) = \{\}$.

We also consider essential v -subgraphs. It is clear that two pre-image trees which agree on some v -subgraph X must also agree on all v -subgraphs contained in X . On the other hand, two pre-image trees may agree on two v -subgraphs X, Y but still differ on the union v -subgraph $X \cup Y$. For example, recall the XML data tree T'_{photo} and its v_{entry} -walks `format` and `section`. Can we say when agreement on two v -subgraphs X, Y forces agreement on their union $X \cup Y$? We found the following condition: Two distinct v -subgraphs X, Y are *v -reconcilable* iff X contains every w -walk in Y or Y contains every w -walk in X , whenever X, Y share some arc (u, w) of frequency other than $?$ and 1 where w is a proper descendant of v . For an example, take the v_{entry} -walks `entrant` and `year`.

Given two v -reconcilable v -subgraphs X, Y and any two pre-image trees p, q we can show that $p_1|_X \doteq p_2|_X$ and $p_1|_Y \doteq p_2|_Y$ iff $p_1|_{X \cup Y} \doteq p_2|_{X \cup Y}$.

Let $\mathbf{E}_T^{\mathfrak{S}}(v)$ be the smallest subset of $\mathfrak{S}_T(v)$ such that the empty v -subgraph and every singleton v -subgraph consisting of a single v -walk belong to $\mathbf{E}_T^{\mathfrak{S}}(v)$ and such that, if two v -subgraphs $X, Y \in \mathbf{E}_T^{\mathfrak{S}}(v)$ are not v -reconcilable then $X \cup Y \in \mathbf{E}_T^{\mathfrak{S}}(v)$. We call the members of $\mathbf{E}_T^{\mathfrak{S}}(v)$ *essential v -subgraphs*. For a set $\mathcal{X}^{\mathfrak{S}}$ of v -subgraphs take all essential v -subgraphs that are contained in some member of $\mathcal{X}^{\mathfrak{S}}$, and among these take only the ones that are maximal with respect to subgraph containment. This gives us $\vartheta(\mathcal{X}^{\mathfrak{S}})$.

There is a practical way of finding all essential v -subgraphs: A *v -unit* is a v -subgraph U that consists of *i*) a single v -walk in which every proper descendant of v has an incoming arc of frequency $?$ or 1 ; or *ii*) all w -walks where w is the proper descendant of v whose incoming arc is the only arc in the path from v to w with frequency $*$ or $+$. Note that each v -walk belongs to exactly one v -unit. We can prove that a v -subgraph is essential iff it is contained in a v -unit.

Given a set \mathcal{X} of v -properties, let $\mathcal{X}^{\mathfrak{A}}$ be the v -ancestors and $\mathcal{X}^{\mathfrak{S}}$ the v -subgraphs in it. Then the translation is $\vartheta(\mathcal{X}) = \vartheta(\mathcal{X}^{\mathfrak{S}}) \cup \vartheta(\mathcal{X}^{\mathfrak{A}})$.

Definition 7.1 (canonical pXFD). A pXFD $v : \mathcal{X} \rightarrow \mathcal{Y}$ is a canonical pXFD iff $\mathcal{X} = \vartheta(\mathcal{X})$ and $\mathcal{Y} = \vartheta(\mathcal{Y})$.

We can prove that a data tree T' satisfies $v : \mathcal{X} \rightarrow \mathcal{Y}$ if and only if T' satisfies $v : \vartheta(\mathcal{X}) \rightarrow \vartheta(\mathcal{Y})$. That is, it is indeed sufficient to study canonical pXFDs.

Construction of two-pre-image data trees. Next, we briefly outline the two-phase approach proposed in Section 5 for constructing a two- v -pre-image data tree which will agree precisely on the v -properties in some given equality set \mathcal{E}_T . For our purposes here, we found an alternative form of input beneficial. For that, an equality set of v -properties is transformed into a corresponding set of v -properties on which the two pre-image trees must *minimally differ*:

Definition 7.2 (non-equality set). Given an equality set of v -properties \mathcal{E}_T , the corresponding non-equality set $\mathcal{N}\mathcal{E}_T$ is given by

$$\mathcal{N}\mathcal{E}_T = \{n \in \check{\mathfrak{A}}_T(v) \mid n \notin \mathcal{E}_T \text{ and } \nexists \text{ a proper ancestor } m \text{ of } n \text{ s.t. } m \notin \mathcal{E}_T\} \\ \cup_{\subseteq - \min} (\{W \in \mathfrak{S}_T(v) \mid W \notin \mathcal{E}_T\})$$

where $\subseteq_{-\min}(\mathcal{S})$ denotes the subset of v -subgraphs belonging to set \mathcal{S} which are minimal with respect to subgraph containment.

Non-equality sets observe some nice characteristics: The members of a non-equality set are pairwise incomparable with respect to ancestor/descendant relationships and subgraph containment. All v -subgraphs in a non-equality set are essential, and a non-empty non-equality set contains precisely one essential v -ancestor with an incoming arc of frequency other than ? and 1.

For the construction under discussion, we look for two pre-image trees p_1, p_2 that differ on a *single essential v -subgraph* $W \in \mathcal{NE}_T$ but agree on every v -subgraph properly contained in W . For a data tree T' in which leaves are assigned the value “0” or “1”, let the *ones-subgraph* X be the largest v -subgraph for which $\text{val}(T'|_B) = \text{“1”}$ iff the v -walk B is contained in X . In the first phase, we can consider each v -unit independently and use the following construction steps:

```

proc construct-pre-image-trees_one-subgraph( $W, U$ )
  Initialise  $p_1^W, p_2^W$  as empty XML trees
  for all (possibly empty)  $v$ -subgraph  $X$  which is contained in  $W$  do
    Create a copy  $c^X$  of  $U$  whose ones-subgraph is  $X$ 
    if  $|X|$  is odd then
       $p_1^W = \text{Merge } c^X \text{ with } p_1^W \text{ on } \eta(U)$ 
    else
       $p_2^W = \text{Merge } c^X \text{ with } p_2^W \text{ on } \eta(U)$ 
    end if
  end for
  return  $p_1^W, p_2^W$ 

```

By “merging” two data trees on some node w we mean that the merged data trees must share every node except their respective pre-images of w and its descendants. Observe that a data tree cannot contain multiple copies of U unless U itself contains more than one walk. If U is a singleton then there is only one copy of U to merge with the empty tree, which trivially returns the copy of U . If U is not a singleton, then it has an identifying node $\eta(U)$ whose incoming arc is the only arc on the path from v to $\eta(U)$ having frequency other than ? and 1 which is shared by all walks in U .

By repeating the previous construction and merging the resulting data trees, we obtain for each v -unit U two U -compatible pre-image trees of v which minimally differ on every v -subgraph in $\mathcal{NE}_T|_U = \subseteq_{-\min}(\{W \in \check{\mathcal{S}}_T(v) \mid W \text{ is contained in } U \text{ and } W \in \mathcal{NE}_T\})$. Note that $\mathcal{NE}_T|_U$ can only have more than one member when U consists of more than one v -walk, and we have again some proper descendant $\eta(U)$ of v with incoming arc of frequency other than ? and 1. Thus, we can merge the pre-image trees $p_i^{W_j}$ ($i = 1, 2, W_j \in \mathcal{NE}_T|_U$) such that they share as many nodes as possible except their pre-images of $\eta(U)$ and its descendants.

Finally, we combine the subtrees constructed for the individual v -units to form a T -compatible data tree containing precisely two pre-image trees which agree on precisely those v -properties belonging to the given equality set \mathcal{E}_T .