Approximate Functional Dependencies for XML Data

Fabio Fassetti and Bettina Fazzinga

DEIS - Università della Calabria
Via P. Bucci, 41C
87036 Rende (CS), Italy
{ffassetti,bfazzinga}@deis.unical.it

Abstract. Functional dependencies (FDs) are an integral part of database theory since they are used in integrity enforcement and in database design. Recently, functional dependencies satisfied by XML data (XFDs) have been introduced. In this work *approximate* functional dependencies that are XFDs approximately satisfied by a considerable part of the XML database are defined and the problem of inferring such XFDs is addressed.

1 Introduction

Traditionally, in order to guarantee data consistency, integrity constraints are essential [1, 14]. Among them, it is widely recognized that functional dependencies (FDs) are the most important and common semantic constraints encountered in databases.

In classical approach FDs are considered to be provided from database designers, but in several areas as data cleaning, data integration and data analysis, they may also be retrieved from the extensional data. The problem of extracting functional dependencies has been deeply addressed for relational databases [4, 11, 13] and recently investigated also with a data mining point of view [12]. Another important goal in the context of using FDs as knowledge to mine from data is to extract FDs that are not satisfied by all the database entries but from the majority of them [9, 10]. This allows the discovery of erroneous or exceptional elements in the data, that is very useful for data cleaning and data integration. Furthermore, this kind of FDs let us to identify constraints very frequent in the database that are meaningful for data analysis, even if they are not valid in the whole database.

Recently integrity constraints have been introduced also for XML data [2, 5–8, 15, 16]. In particular, in [2, 15] two definitions of XML functional dependencies (XFDs) are given. Both these definitions treat the equality between two XML elements as equality between their identifiers and they do not use the subtrees equivalence. Even if this approach can be useful for a database designer and for normalizing an XML database, it does not fit well when the FDs are unknown, and the goal is to extract them from the database. To this aim, in [16] a novel

Y. Ioannidis, B. Novikov and B. Rachev (Eds.): Local Proceedings of ADBIS 2007, pp. 86-95

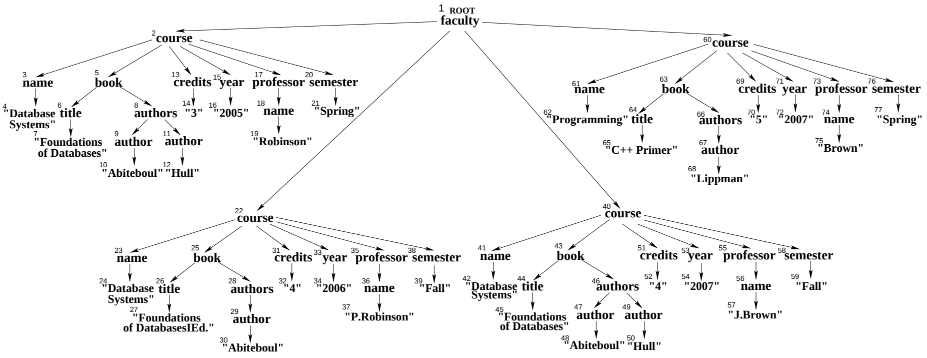

Fig. 1: *UnivInfo* XML Tree

definition for XML functional dependency, using subtree equivalence as element equivalence, is proposed and the problem of inferring XFDs from XML data is addressed. Nevertheless, authors do not deal with the problem to discover approximate XFDs.

The complex structure of XML data adds an interesting facet about the satisfaction of XFDs. In fact, whereas the satisfaction of an FD in a relational database is checked comparing values contained in the tuple of the database, the satisfaction of an XFD implies the tree comparison since each "tuple" of an XML database is composed by trees. Since tree structure is complicated and it can contain several subtrees, it can happen that two trees describe the same object, but their shape and content are slightly different. In these cases, these trees invalidate the satisfaction of an XFD because their are not perfectly equal. Instead, it is very interesting and useful to evaluate how much trees are similar by means of a proper measure and then, if they are judged "enough" similar, the XFD is to be considered *approximately* satisfied. For example, consider the XML tree shown in Figure 1. Obviously, trees rooted at "book" elements 5 and 25 identify the same book, but they are not perfectly the same. In this case, checking the satisfaction of the XFD claiming that *the name of a course implies the text book* yields a negative answer using the approach proposed in [16], but in fact the XFD is satisfied.

Furthermore, in many cases, extracting functional dependencies that do not hold in the whole database, but in the majority of it, is very useful, as stated previously. For example, consider the XML tree in Figure 1 and the XFD claiming that *the course name implies the number of credits*. Such an XFD does not hold in the whole database, since for the course "Database Systems", the number of credits is "3" in "2005" and "4" in "2006". Therefore, using approach proposed in [16], such an XFD is not inferred, but, since the XFD almost always holds in the database, it is interesting and then should be extracted.

Finally, if no approximation is employed in checking XFD satisfaction, it could happen that some XFDs that in fact do not hold are extracted. For example, consider the XML tree shown in Figure 1 and the XFD claiming that *professor implies the semester*. Since all the professors are slightly different from one another, this XFD holds if approach proposed in [16] is considered. Nevertheless, professor rooted at 17 is much similar to professor rooted at 35, and professor rooted at 55 is similar to professor rooted at 73, but since semesters are different the XFD does not hold and it should not be extracted.

The aim of this work is the discovery of approximate functional dependencies on XML documents, namely the set of XFDs that are approximately satisfied by a given collection of XML data. We introduce a notion of approximate satisfaction of an XFD (σ-*approximation*) based on the dissimilarity measure between trees and we use this new concept for discovering XFDs σ-approximately satisfied by a percentage of "tuples" of the XML database specified in advance. To the best of our knowledge this is the first work dealing with the discovery of approximate XFDs.

The rest of the paper is planned as follows. Section 2 provides some preliminary definitions and the notation used in the paper. Section 3 presents the definition of approximate XFDs. Section 4 addresses the inference problem. Finally, Section 5 draws the conclusions.

2 Preliminaries

Let $\mathcal{L}$ be a countably infinite set of labels, $\mathcal{S}$ be a countably infinite set of strings and let $\epsilon \notin \mathcal{S}$ denote the empty string.

Definition 1 (XML Tree). *An* XML tree *is a quadruple* $\mathcal{T} = \langle r, \mathcal{N}, \mathcal{V}, \lambda \rangle$*, where* $\mathcal{N}$ *denotes the set of nodes,* $r \in \mathcal{N}$ *denotes the root of* $\mathcal{T}$*,* $\mathcal{V} \subseteq \mathcal{N} \times \mathcal{N}$ *denotes the set of edges and* $\lambda : \mathcal{N} \rightarrow \mathcal{L} \cup \mathcal{S}$ *is a function s.t.* $\forall\ n_i \in \mathcal{N}$*,*

- $\lambda(n_i) \in \mathcal{L}$*, if* $\exists\ (n_i, n) \in \mathcal{V}$*;* *(*$n_i$ *is called* element node*)*
- $\lambda(n_i) \in \mathcal{S}$*, otherwise* *(*$n_i$ *is called* text node*)*

In the rest of the paper, we assume that the XML tree is not recursive, that is there are no two nodes having the same label s.t. one of them is descendant of the other one.

Definition 2 (Path). *A* path *is an expression of the form* $/l_1/\ldots/l_k$*, where* $k \geq 1$ *and*

- $l_i \in \mathcal{L}$*,* $i \in [1..k-1]$*;*
- $l_k \in \mathcal{L} \cup \{\#text\}$

Definition 3 (Path Prefix). *Given two paths* $p = /l_1/\ldots/l_k$ *and* $p' = /l'_1/\ldots/l'_h$*,* p *is a prefix of* p' *iff* $k \leq h \wedge l_i = l'_i\ \forall i \in [1, \ldots, k]$*.*

Definition 4 (Path Instance). *Given a path* $p = /l_1/\ldots/l_k$ *and an XML tree* $\mathcal{T} = \{r, \mathcal{N}, \mathcal{V}, \lambda\}$*,* p *matches on* $\mathcal{T}$ *iff there exists* $\{n_1, \ldots, n_k\} \subseteq \mathcal{N}$ *s.t. the following conditions hold:*

- $\lambda(n_i) = l_i, \forall i \in [1..k-1]$

- $\begin{cases} \lambda(n_k) = l_k & \textit{if } l_k \in \mathcal{L} \\ \lambda(n_k) \in \mathcal{S} & \textit{if } l_k = \#text \end{cases}$
- $(n_i, n_{i+1}) \in \mathcal{V}, \forall i \in [1..k-1]$

In such a case, we say that $\pi = /n_1/\ldots/n_k$ *is a* path instance *of* p *in* $\mathcal{T}$*, and* n_k *is the* ending node *of* π *in* $\mathcal{T}$*.*

For instance, considering Figure 1, $/faculty/course$, $/professor/name$ are paths, $/faculty/course/professor$ is a prefix of $/faculty/course/professor/name$, and $\{/1/2/15\}$ is a path instance of $/faculty/course/year$, whereas $\{/1/2/15/16\}$ is a path instance of $/faculty/course/year/\#text$.

Definition 5 (Ending Nodes Set). *Let* $\mathcal{T}$ *be an XML tree,* $p = /l_1/\ldots/l_k$ *be a path, and* $\Pi = \{\pi_1, \ldots, \pi_h\}$ *be the set of all the path instances of* p *in* $\mathcal{T}$*. The ending nodes set* $\mathcal{EN}(p) = \{m_1, \ldots, m_h\}$ *is the set of nodes of* $\mathcal{T}$ *s.t. each* m_i *is the ending node of* π_i *in* $\mathcal{T}$*.*

For example, consider the path $p = /faculty/course/name$ the ending nodes set of p is $\mathcal{EN}(p) = \{3, 23, 41, 61\}$.

Definition 6 (Repeated Path). *Let* p *and* p' *be two paths.* p' *is said to be a repeated path for* p *if* p *is a prefix of* p' *and there exists at least one node* $n \in \mathcal{EN}(p)$ *that is ancestor of at least two ending nodes of* p'*.*

For example, $/faculty/course/book/authors/author$ is a repeated path for $/faculty/course/book/authors$ since there exists node $8 \in \mathcal{EN}(/faculty/course/book/authors)$ and there exist node 9 and 11, belonging to $\mathcal{EN}(/faculty/course/book/authors/author)$, both descendants of 8.

Definition 7 (Root Path). *Given an XML tree* $\mathcal{T} = \langle r, \mathcal{N}, \mathcal{V}, \lambda \rangle$ *and a path* $p = /l_1/\ldots/l_h$*,* p *is a* root path *w.r.t.* $\mathcal{T}$*, iff* $l_1 = \lambda(r)$*.*

In the sequel of the paper, whenever we refer to *paths* we mean *root paths.*

In order to handle missing elements (*nulls*), the completion of XML trees is needed. Intuitively, given an XML tree $\mathcal{T}$ and a set of paths, an XML extended tree of $\mathcal{T}$ w.r.t. the set of paths is an XML tree completed with nulls so that each path of the set is matched.

Before defining the extended tree, we give our notion of XML tree containment.

Definition 8 (Tree containment). *Given an XML tree* $\mathcal{T} = \langle r, \mathcal{N}, \mathcal{V}, \lambda \rangle$*, an XML tree* $\mathcal{T}' = \langle r, \mathcal{N} \cup \Upsilon, \mathcal{V}', \lambda' \rangle$ *contains* $\mathcal{T}$ $(\mathcal{T} \sqsubseteq \mathcal{T}')$*, if the following conditions hold:*

- $\Upsilon = \{\bot_1, \ldots, \bot_k\}$ *is a set of marked nulls.*
- $\mathcal{V}' \supseteq \mathcal{V}$
- $\lambda'(n) = \lambda(n), \forall n \in \mathcal{N}$
- $\lambda'(m) \in \mathcal{L} \cup \{\epsilon\}, \forall m \in \Upsilon$

Last condition indicates that if a marked null represents an element then its label must belong to $\mathcal{L}$, otherwise its label must be empty.

Definition 9 (Extended Tree). *Let $\mathcal{T}$ be an XML tree and $\mathcal{P}$ the set of all paths of $\mathcal{T}$. An XML tree $\mathcal{T}' = \xi(T)$ is an* extended tree *of $\mathcal{T}$ if the following conditions hold:*

- $\mathcal{T} \sqsubseteq \mathcal{T}'$
- *for each path $p \in \mathcal{P}$ if there exists a path instance $\pi = /n_1/\ldots/n_m$ of q in $\mathcal{T}'$, where q is a prefix of p and $q \neq p$, then there exists a path instance $\pi' = /n'_1/\ldots/n'_n$ of p in $\mathcal{T}'$ s.t. $n_i = n'_i, i \in [1..m]$.*

$\mathcal{T}' = \langle r, \mathcal{N} \cup \Upsilon', \mathcal{V}', \lambda' \rangle$ is said to be minimal *if there not exist an extended tree $\mathcal{T}'' = \langle r, \mathcal{N} \cup \Upsilon'', \mathcal{V}'', \lambda'' \rangle$ of $\mathcal{T}$ such that $|\Upsilon''| < |\Upsilon'|$.*

3 Approximate XML Functional Dependencies

In order to define approximate XFDs, two kinds of approximations are taken in account. The first one (*σ-approximation*) concerns the structural and content similarity between "tuples" (trees) involved in the satisfaction of XFDs, whereas the second one concerns the percentage of the whole database that σ-approximately satisfies the functional dependencies. Therefore, *approximate XFDs* are XFDs σ-approximately satisfied by a large part of a given XML document. Intuitively, if two different instances of the left part of an XFD are quite similar, then they can be considered as describing the same real-world entity. Thus, the correspondent instances of the right part have to represent the same entity, and then they must be similar, too.

The approach followed in this work needs neither a DTD nor an XML schema defined on the document. Then, in order to single out "tuples" on which FDs have to be verified, an approach similar to that proposed in [15] is exploited.

3.1 XFD Definitions

Firstly, the formal definition of XFD is given.

Definition 10 (XML Functional Dependency). *Given an XML tree T, and two sets of paths $\{p_1, \ldots, p_m\}$ and $\{q_1, \ldots, q_n\}$ of T, an XML functional dependency (XFD) is an expression of the form $\mathsf{p_1, \ldots, p_m \rightarrow q_1, \ldots, q_n}$. The set $\{p_1, \ldots, p_m\}$ (resp. $\{q_1, \ldots, q_n\}$) is called* left part *(resp.* right part*) of the XFD.*

Since $\mathsf{p_1, \ldots, p_m \rightarrow q_1, \ldots, q_n}$ is equivalent to the set of XFDs

$$\{\mathsf{p_1, \ldots, p_m \rightarrow q_1}; \quad \ldots \quad ; \mathsf{p_1, \ldots, p_m \rightarrow q_n}\}$$

w.l.o.g., in the following, we deal with XFDs whose right part contains only one path.

In order to establish which nodes have to be considered belonging to the same "instance" of an XFD, we recall the definition of *closest node* as proposed in [15]. Note that the definition is in a little different form to match the setting of this work.

Definition 11 (Closest Node). *Given two paths p' and p'', let p be the longest common prefix of p' and p''. Let n' and n'' be two nodes s.t. $n' \in \mathcal{EN}(p')$ and $n'' \in \mathcal{EN}(p'')$. n' is a* closest node *of n'' ($n' \parallel n''$) iff there exists a node $n \in \mathcal{EN}(p)$ s.t. n is an ancestor of both n' and n''.*

It follows that the above relation is symmetric, reflexive and intransitive. Note that, in general, for each node can exist more than one closest node for the same path.

Example 1. Consider the $UnivInfo$ XML database in Figure 1 and paths $p' = /faculty/course/name$ and $p'' = /faculty/course/professor/name$. The longest common prefix between p' and p'' is $p = /faculty/course$. Consider nodes $n' = 3 \in \mathcal{EN}(p')$ and $n'' = 18 \in \mathcal{EN}(p'')$. n' is a closest node of n'' since there exists a node $n = 2 \in \mathcal{EN}(p)$ that is a common ancestor of both 3 and 18. Let's now consider node $n''' = 36 \in \mathcal{EN}(p'')$. n' is not a closest node of n''' since there not exists a node in $\mathcal{EN}(p)$ that is a common ancestor of both 3 and 36.

In order to group nodes representing an "instance" of an XFD we introduce the *matching node set*, that is formed by an ending node for each path contained in the XFD. Each node in the set is closest of all other nodes in the set.

Definition 12 (Matching Node Set). *Given an XML tree $\mathcal{T}$ and a set of paths $\mathcal{P} = p_1, \ldots, p_m$, a matching node set of $\mathcal{P}$ in $\mathcal{T}$ is a set $\mathcal{Mns}(P) = \{z_1, \ldots, z_m\}$ of nodes of $\xi(\mathcal{T})$ s.t. for each $i, j \in [1..m]$, $z_i \in \mathcal{EN}(p_i) \wedge z_i \parallel z_j$.*

Note that, in the above definition, the extended tree of $\mathcal{T}$ is employed. This is needed in order that each ending node of any path in $\mathcal{P}$ belongs to at least one matching node set. Indeed, if an incomplete tree were used, it could happen that for an ending node n of a path $p \in \mathcal{P}$ there not exists an ending node n' of an other path $p' \in \mathcal{P}$ s.t. $n \parallel n'$. Thus, n would not belong to any matching node set.

Given a set of paths $\mathcal{P}$, a matching node set M of $\mathcal{P}$ and a path $p \in \mathcal{P}$, we denote as M^p the node of M belonging to $\mathcal{EN}(p)$.

In order to deal with approximate XFDs, we need a way to evaluate when two distinct "instances" of an XFD have to be considered similar. Since the ending nodes are trees, we employ the tree edit distance [3] as similarity measure. Roughly, this kind of distance between two trees is based on the computation of sequences of edit operations needed to transform a tree in the other. The edit operations are *node insertions*, *node deletions* and *node renamings*. Each operation is associated with a cost and the cost of a sequence of edit operations is the sum of the costs of each operation composing the sequence. The tree edit distance is defined as the cost of the sequence having minimum cost.

Definition 13 (Tree Distance). *Let $\mathcal{T}$ and $\mathcal{T}'$ be two XML trees and φ be a cost function. An* edit script *$\mathcal{S}$ between $\mathcal{T}$ and $\mathcal{T}'$ is a sequence of edit operations turning $\mathcal{T}$ into $\mathcal{T}'$. The cost of $\mathcal{S}$ is the sum of the costs of the operations in $\mathcal{S}$ employing φ. The* tree distance *between $\mathcal{T}$ and $\mathcal{T}'$ is*

$$\delta(\mathcal{T},\mathcal{T}') = \frac{\mathsf{cost}(\mathcal{S}^*)}{|\mathcal{T}| + \mathsf{ins}(\mathcal{T},\mathcal{T}')}$$

where $\mathcal{S}^*$ is the edit script having minimum cost, cost *indicates the cost of the edit script and* ins *denotes the number of node insertions necessary to transform $\mathcal{T}$ in $\mathcal{T}'$.*

Note that the edit distance between two trees $\mathcal{T}$ and $\mathcal{T}'$ is normalized w.r.t. the sum of the size of $\mathcal{T}$ and the number of node insertions necessary to transform the former tree in the latter one.

Now, we employ the above concepts for measuring the distance between matching node sets.

Definition 14 ($\mathcal{M}ns$ Distance). *Let $\mathcal{P}$ and $\mathcal{P}' \subseteq P$ be two sets of paths, and let M and Q be two matching node sets of $\mathcal{P}$. The distance between M and Q w.r.t. $\mathcal{P}'$ is*

$$\mu_{P'}(M,Q) = max_{p\in\mathcal{P}'}(\delta(M^p,Q^p))$$

3.2 Approximate Satisfaction of XFDs

Now, XFDs approximately satisfied by an XML tree can be formally defined.

Definition 15 (σ-approximation). *Let $f : P \rightarrow q$ be an XFD, $\mathcal{T}$ be an XML tree, R be a set of matching node sets of $(P \cup \{q\})$ in $\mathcal{T}$ and σ be a dissimilarity threshold. R σ-approximately satisfies f ($R \models_\sigma f$) if, for every two distinct matching node sets $M_1, M_2 \in R$, it holds that*

$$\mu_P(M_1,M_2) \leq \sigma \Rightarrow \mu_{\{q\}}(M_1,M_2) \leq \sigma \qquad (1)$$

In the following a set of matching node sets that σ-approximately satisfies an XFD is called *solving set* of the XFD.

Example 2. Consider the XML tree in Figure 1, the XFD $f : /faculty/course/name \rightarrow /faculty/course/book$ and $\sigma = 0.3$. Consider the set of matching node sets of $\{/faculty/course/name, /faculty/course/book\}$

$$R = \{\{3,5\},\{23,25\},\{41,43\},\{61,63\}\}$$

The pairs judged to be similar on the left part of the XFD are $(\{3,5\},\{23,25\})$, $(\{3,5\},\{41,43\})$. Since these pairs also satisfy the right side of implication (1) (the distance between node 5 and 25, between node 5 and 43 and between node 25 and 43 is $0.29 \leq \sigma$), R σ-approximately satisfies f.

Let's now consider $f' : /faculty/course/professor \rightarrow /faculty/course/semester$ and the set of matching node sets of $\mathcal{P} = \{/faculty/course/professor, /faculty/course/semester\}$

$$R = \{\{17,20\},\{35,38\},\{55,58\},\{73,76\}\}$$

The pair ({17,20},{35,38}) satisfies the left side of implication (1), but it does not satisfy the right side of implication (1). Then, R does not σ-approximately satisfies f.

Given an XFD f, several solving sets can be extracted from a document and some of these, called *maximal solving sets*, have maximum cardinality. More formally, a solving set R is said *maximal* if there not exists another solving set R' s.t. $|R| < |R'|$. Note that can exist more than one maximal solving set. The value of the cardinality of a maximal solving set is exploited to compute the percentage of the whole database that satisfies f.

Now, the formal definition of approximate XFD satisfaction can be given.

Definition 16 (Approximate Satisfaction of an XFD). *Let $f : P \to q$ be an XFD, $\mathcal{T}$ be an XML tree, σ be a dissimilarity threshold and θ be a satisfaction threshold. Let R be the set of all the matching node sets of $P \cup \{q\}$ in $\mathcal{T}$ and R' be a* maximal solving set*. $\mathcal{T}$ (σ, θ)-approximately satisfies f ($\mathcal{T} \models^{\theta}_{\sigma} f$) if $\frac{|R'|}{|R|} \geq \theta$.*

Example 3. Consider XML tree in Figure 1 again, suppose $\sigma = 0.3$ and $\theta = 0.7$ and consider the XFD

$$f : /faculty/course/name \to /faculty/course/credits$$

The matching node sets of $\{/faculty/course/name\} \cup \{/faculty/course/credits\}$ are

$$M_1 = \{3, 13\}, M_2 = \{23, 31\}, M_3 = \{41, 51\}, M_4 = \{61, 69\}$$

Let $R = \{M_1, M_2, M_3, M_4\}$ be the set of all the matching node sets. In order to valuate the confidence of f we have to compute the solving sets. It can be easily seen that the subsets $R' = \{M_2, M_3, M_4\}$ and $R'' = \{M_1, M_4\}$ are solving sets since they satisfy f. Note that, for example, M_1 and M_2 can not belong to the same solving set since their distance w.r.t. $/faculty/course/name$ is $0 < \sigma$ but their distance w.r.t. $/faculty/course/credits$ is $1 > \sigma$, and then they violate f. The maximal solving set is R'. Since $|R'|/|R| = 3/4 = 0.75$, then $\mathcal{T}$ (σ, θ)-*approximately* satisfies f.

4 Inferring XFDs

In this section, we formally define the problem of inferring approximate XFDs from an XML document.

Not all the XFDs approximately satisfied by an XML document need to be inferred, since some of them are deducible from others without any computations and some others are not interesting.

Consider, for example, the XML tree $\mathcal{T}$ in Figure 1, $\sigma = 0.3$, $\theta = 0.7$, and the following subset of XFDs (σ, θ)-*approximately* satisfied by $\mathcal{T}$:

$$\begin{aligned}
\mathcal{F} = \{&\mathsf{/faculty/course/name} \to \mathsf{/faculty/course/book}; && (1)\\
&\mathsf{/faculty/course/name, /faculty/course/year} \to \mathsf{/faculty/course/book}; && (2)\\
&\mathsf{/faculty/course/name, /faculty/course/year} \to \mathsf{/faculty/course/year}; && (3)\\
&\mathsf{/faculty/course/name} \to \mathsf{/faculty/course/book/title}; && (4)\\
&\mathsf{/faculty/book} \to \mathsf{/faculty/course/name}; && (5)\\
&\mathsf{/faculty/book/title} \to \mathsf{/faculty/course/name;}\} && (6)
\end{aligned}$$

It is easily to see that only (1) and (5) are interesting and then the others should not be extracted. Indeed, (2) is negligible since it carries an information that is already mined by means of (1). (3) is always satisfied, since the path used as right part is included in the set of paths used as left parts. Finally, (4) and (6) are not interesting, since the information carried by them is embedded in (1) and (5) respectively.

Thus, to avoid functional dependencies that are trivially satisfied or non-significative, we give a definition of *meaningless XML functional dependencies.*

Definition 17 (Meaningless XFDs). *Let $\mathcal{T}$ be an XML tree and $f : \{p_1, \ldots, p_k\} \rightarrow q$ be an XFDs. f is said to be trivial if one of the following conditions holds:*

1. *there exists $i \in [1..k]$ s.t. $p_i = q$.*
2. *there exist $i, j \in [1..k], j \neq i$ s.t. p_i is a prefix of p_j;*

In the first case, the XFD is always satisfied, thus is needless to be considered in the XFD inference phase. In the second case, the XFD is not significative since the information contained in p_j is redundant.

In order to reduce the number of XFDs to be examined, we introduce a *precedence* relation between XFDs.

Definition 18 (XFDs Precedence). *Let $\mathcal{T}$ be an XML tree and $f : \{p_1, \ldots, p_h\} \rightarrow q$ and $f' : \{p'_1, \ldots, p'_k\} \rightarrow q'$ be two XFDs. f precedes f' ($f \prec f'$) if one of the following conditions holds:*

1. *$q = q'$, $k > h$ and $\forall i \in [1..h], p_i = p'_i$;*
2. *q is a prefix of q', $k = h$ and $\forall i \in [1..h] p_i$ is a prefix of p'_i.*

In particular, if f is (σ, θ)-*approximately* satisfied then f' is also (σ, θ)-*approximately* satisfied in the first case, whereas f' is needless to be inferred in the second case.

Then, the goal is to extract a set of functional dependencies that is minimal in according to the following definition:

Definition 19. *Let $\mathcal{F}$ be a set of non-meaningless XFDs, $\mathcal{F}$ is minimal if*

$$\forall f \in \mathcal{F}, \nexists f' \in \mathcal{F} \text{ such that } f' \neq f \wedge f' \prec f$$

Now, the task of inferring XFDs approximately satisfied by an XML tree can be formally presented.

Definition 20. *Given an XML tree $\mathcal{T}$, a dissimilarity threshold σ and a satisfaction threshold θ, the XFD inference problem is the extraction from $\mathcal{T}$ of the minimal set $\mathcal{F}$ of XFDs, such that $\forall f \in \mathcal{F}$ f is (σ, θ)-approximately satisfied by $\mathcal{T}$ ($\mathcal{T} \models^{\theta}_{\sigma} \mathcal{F}$).*

Due to lack of space, we omit the algorithm solving the *XFD inference problem*.

5 Conclusions

In this paper we defined approximate XFDs, namely XFDs approximately satisfied (according to tree similarity) by a certain part of an XML document. Furthermore, we addressed the problem of inferring approximate XFDs from data.

References

1. S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases.* Addison-Wesley, 1995.
2. M. Arenas and L. Libkin. A normal form for xml documents. *TODS*, 29:195–232, 2004.
3. Philip Bille. A survey on tree edit distance and related problems. *TCS*, 337(1-3):217–239, 2005.
4. D. Bitton, J. Millman, and S. Torgersen. A feasibility and performance study of dependency inference. In *ICDE*, pages 635–641, 1989.
5. T. Bray, J. Paoli, and C. M. Sperberg-McQueen. Extensible Markup Language (XML) 1.0 - W3C recommendation. Technical Report REC-xml-19980210, 1998.
6. P. Buneman, S. B. Davidson, W. Fan, C. S. Hara, and W. C. Tan. Reasoning about keys for xml. *IS*, 28(8):1037–1063, 2003.
7. P. Buneman, W. Fan, and S. Weinstein. Path constraints in semistructured databases. *JCSS*, 61(2):146–193, 2000.
8. W. Fan and J. Siméon. Integrity constraints for xml. In *PODS*, pages 23–34, 2000.
9. Y. Huhtala, J. Kärkkäinen, P. Porkka, and H. Toivonen. Tane: An efficient algorithm for discovering functional and approximate dependencies. *Comput. J.*, 42(2):100–111, 1999.
10. J. Kivinen and H. Mannila. Approximate inference of functional dependencies from relations. *TCS*, 149:129–149, 1995.
11. H. Mannila and K.-J. Räihä. Algorithms for inferring functional dependencies from relations. *DKE*, 12(1):83–99, 1994.
12. N. Novelli and R. Cicchetti. Functional and embedded dependency inference: a data mining point of view. *IS*, 26(7):477–506, 2001.
13. I. Savnik and P. A. Flach. Bottom-up induction of functional dependencies from relations. In *AAAI-93 Workshop: Knowledge Discovery in Databases*, pages 174–185, 1993.
14. J. D. Ullman. *Principles of Database Systems, 2nd Edition.* Computer Science Press, 1982.
15. M. W. Vincent, J. Liu, and C. Liu. Strong functional dependencies and their application to normal forms in xml. *TODS*, 29(3):445–462, 2004.
16. C. Yu and H. V. Jagadish. Efficient discovery of xml data redundancies. In *VLDB*, pages 103–114, 2006.