Reconciling Inconsistent Data in Probabilistic XML Data Integration

Tadeusz Pankowski[1,2]

[1] Institute of Control and Information Engineering,
Poznań University of Technology, Poland
[2] Faculty of Mathematics and Computer Science,
Adam Mickiewicz University, Poznań, Poland
tadeusz.pankowski@put.poznan.pl

Abstract. The problem of dealing with inconsistent data while integrating XML data from different sources is an important task, necessary to improve data integration quality. Typically, in order to remove inconsistencies, i.e. conflicts between data, data cleaning (or repairing) procedures are applied. In this paper, we present a probabilistic XML data integration setting. A probability is assigned to each data source and its probability models the *reliability level* of the data source. In this way, an answer (a tuple of values of XML trees) has a probability assigned to it. The problem is how to compute such probability, especially when the same answer is produced by many sources. We consider three semantics for computing such probabilistic answers: *by-peer*, *by-sequence*, and *by-subtree* semantics. The probabilistic answers can be used for resolving a class of inconsistencies violating XML functional dependencies defined over the target schema. Having a probability distribution over a set of conflicting answers, we can choose the one for which the probability of being correct is the highest.

1 Introduction

In general, in data integration systems (especially in P2P data management [12,13]) violations of consistency constraints cannot be avoided [10,15]. Data could violate consistency constraints defined over the target schema, although it satisfies constraints defined over source schemas considered in separation. In the paper we focus on XML functional dependencies as constraints over XML schemas. From a set of inconsistent values violating the functional dependency we choose one which is most likely to be correct. The choice is based on probabilities of data. We propose a model of calculating such probabilities using the reliability levels assigned to data sources.

Related Work. Dealing with inconsistent data is the subject of many work known as data cleaning [14] and consistent query answering in inconsistent databases [2]. There are two general approaches to resolve conflicts in inconsistent databases [4,8,9]: (1) the user provides a procedure deciding how the conflicts should be resolved; (2) some automatic procedures may be used – the

A. Gray, K. Jeffery, and J. Shao (Eds.): BNCOD 2008, LNCS 5071, pp. 75–86, 2008.

procedures can be based on timestamps (outdated data may be removed from consideration) or reliability of data (each conflicting data has a probability of being correct assigned to it). A model based on reliabilities of data sources was discussed in [16] and was used for reconciling inconsistent updates in collaborative data sharing. In [6], authors develop a model of probabilistic relational schema mappings. Because of the uncertainty about which mapping is correct, all the mappings are considered in query answering, each with its own probability. Two semantics for probabilistic data are proposed in [6]: *by-table* and *by-sequence* semantics. Probabilities associated to data are then used to rank answers and to obtain top-k answers to queries in such a setting.

In this paper, we discuss a probabilistic XML data integration setting, where the probability models *reliability levels* of data sources. Based on these we calculate probabilities associated with answers (probabilistic answers) to queries over the target schema. We propose three semantics for producing probabilistic answers: *by-peer*, *by-sequence* (of peers), and *by-subtree* semantics. Two first of them are based on *by-table* and *by-sequence* semantics proposed in [6], but the interpretation of probabilistic mappings as well as data integration settings are quite different.

The main novel contribution of this paper is the introduction of the *by-subtree* semantics. This semantics takes into account not only sources where the answer comes from, but also contexts in which it occurs in data sources. Thanks to this, the method has the advantage over other methods because the computation of the probability is more sensitive to contexts of data in interest.

In Section 2 we introduce a motivating example and illustrate basic ideas of reconciling inconsistent data in a data integration scenario. We show the role of XML functional dependencies and probabilistic answers in reconciliation of inconsistent data. In Section 3 we discuss XML schemas and XML data (XML trees). Schema mappings and queries for XML data integration are described in Section 4 and Section 5, respectively. In Section 6, schema mappings are generalized to *probabilistic* schema mapping. They are used to define probabilistic answers to queries. Section 7 concludes the paper.

2 Reconciliation of Inconsistent Data

To illustrate our approach, let us consider Figure 1, where there are three peers P_1, P_2, and P_3 along with schema trees, S_1, S_2, S_3, and schema instances I_1, I_2, and I_3, respectively.

Over S_3 the following XML functional dependency (XFD) [1] can be defined

$$/authors/author/book/title \rightarrow /authors/author/book/year, \tag{1}$$

meaning that a text value (a tuple of text values) of the left-hand path (tuple of paths) uniquely determines the text value of the right-hand path. Let J be an instance of S_3. If in J there are two subtrees of type $/authors/author/book$

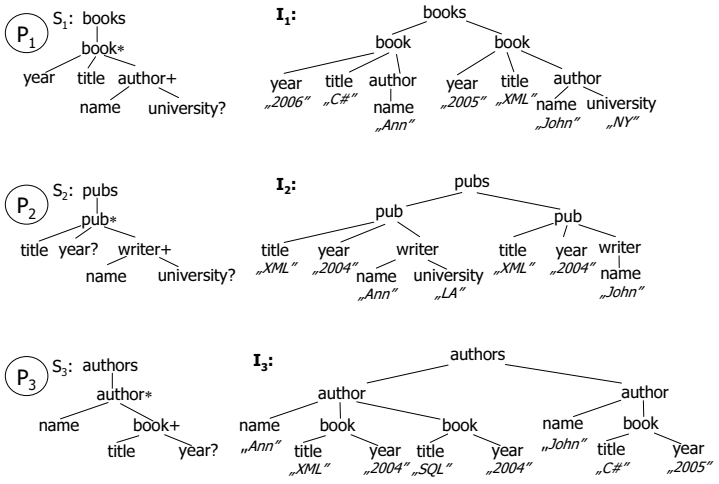

Fig. 1. XML schema trees S_1, S_2, S_3, and their instances I_1, I_2 and I_3, located in peers P_1, P_2, and P_3

which have equal values of *title*, say t, and different values of *year*, say y_1 and y_2, then we say that the set C_t

$$C_t = \{(/authors/author/book/title : t, /authors/author/book/year : y_1), \\ (/authors/author/book/title : t, /authors/author/book/year : y_2)\}$$

is inconsistent with respect to the XFD (1). Further on, the paths labeling values will be omitted, so we will write $C_t = \{(t, y_1), (t, y_2)\}$.

Our aim is to choose such a tuple $(t, y) \in C_t$, that can be treated as the most reliable of all tuples belonging to the inconsistent set C_t. The process of selecting such a tuple is called *reconciliation of inconsistent data*. To this order we will consider three different ways for computing probabilities for any tuple belonging to C_t, where the probability reflects trustworthiness of being correct for the corresponding tuple. Finally, a tuple with the highest probability will be selected.

To build a *probabilistic XML data integration setting*, it will be assumed that a numeric *reliability level* [16] is assigned to every peer's partner and the following *trust policy* is applied:

1. A vector $r_1, ..., r_n$ of reliability levels is assigned to source schemas $S_1, ..., S_n$, with respect to the target schema T, $\Sigma_{i=1}^{n} r_i = 1$. A value r_i is treated as the trustworthiness of data obtained from the source S_i.
2. Reliability level will be understood as *probability* which will be assigned to the mapping m_i from a source schema S_i to the target schema T. In this way we can say about *probabilistic schema mappings*.

Now, assume that we are interested in all pairs $(title, year)$ and that the appropriate query q has been issued against the schema S_3 on the peer P_3.

Table 1. Answers to the query q

S_1 (0.5)	$(C\#, 2006), (XML, 2005)$
S_2 (0.2)	$(XML, 2004), (XML, 2004)$
S_3 (0.3)	$(XML, 2004), (SQL, 2004), (C\#, 2005)$

Assume that reliability levels of sources S_1, S_2, and S_3 are 0.5, 0.2, and 0.3, respectively. In Table 1 there are answers returned from the three sources.

We have seven answers, where some of them are duplicated. The answers can be clustered into the following sets:

$$\begin{aligned} C_{XML} &= \{(XML, 2005), (XML, 2004)\}, \\ C_{C\#} &= \{(C\#, 2006), (C\#, 2005)\}, \\ C_{SQL} &= \{(SQL, 2004)\}, \end{aligned}$$

where C_{XML} and $C_{C\#}$ are inconsistent. Thus, we have to decide which of two answers $(XML, 2005)$ or $(XML, 2004)$ is more *certain*, similarly for $(C\#, 2006)$ and $(C\#, 2005)$. As the measure of uncertainty we assign probabilities to answers, where probabilities are calculated using *reliability levels* of data sources.

To calculate probability of data in the target instance, the following three semantics will be discussed: *by-peer*, *by-sequence*, and *by-subtree* semantics.

By-peer semantics.

Any mapping is considered as a separate event in the space of elementary events. Then probability of data depends only on the peer (data source) where the data comes from. If the same data comes from many sources then its probability is the sum of probabilities of these sources. In [6] this way of creating probabilistic data is referred to as *by-table* semantics.

In Table 3 we can find probabilities of answers according to the *by-peer* semantics. Note that the fact that the tuple $(XML, 2004)$ is returned two times from S_2, has no impact on the final probability of this tuple.

By-sequence semantics.

In this approach, any sequence of mappings (of a given length) is considered as a separate event in the space of elementary events. Then probability of data depends not only on the mapping creating the data but also on the *context in which it is created*. In [6], this way of creating probabilistic data is called *by-sequence* semantics.

In Table 2 there are sequences of mappings of length 2, where a sequence (m_i, m_j) maps an instance, (I_i, I_j) of the pair of schemas (S_i, S_j) into an instance J of the schema S_3, $1 \leq i, j \leq 3$. The probability of each sequence is the multiplication of probabilities of composing mappings, e.g. $Prob(m_1, m_3) = Prob(m_1) * Prob(m_3) = 0.15$. Using probabilities of sequences, we can determine probabilities of answers to q. The probability of an answer is the sum of

Table 2. Calculation of probabilities for $(XML, 2004)$ and $(XML, 2005)$ in the *by-sequence* and *by-subtree* semantics

Seq	*Prob*	$(XML, 2004)$ *by-sequence*	$(XML, 2004)$ *by-subtree*	$(XML, 2005)$ *by-sequence*	$(XML, 2005)$ *by-subtree*
(m_1, m_1)	0.25	N	N	Y	Y
(m_1, m_2)	0.10	Y	Y	Y	N
(m_1, m_3)	0.15	Y	N	Y	N
(m_2, m_1)	0.10	Y	Y	Y	Y
(m_2, m_2)	0.04	Y	Y	N	N
(m_2, m_3)	0.06	Y	Y	N	N
(m_3, m_1)	0.15	Y	Y	Y	Y
(m_3, m_2)	0.06	Y	Y	N	N
(m_3, m_3)	0.09	Y	Y	N	N

Table 3. Probabilities of answers to q in three semantics

Tuple	*By-peer*	*By-sequence*	*By-subtree*
$(C\#, 2006)$	0.5	0.75	0.5
$(XML, 2005)$	0.5	0.75	0.5
$(XML, 2004)$	0.5	0.75	0.6
$(SQL, 2004)$	0.3	0.51	0.3
$(C\#, 2005)$	0.3	0.51	0.3

probabilities of these sequences which return the answer (denoted in the relevant columns in Table 2 by Y). Observe that, like in the case of *by-peer* semantics, the probability of an answer does not depend on the number of occurrences of the answer in the source.

By-subtree semantics.

In this method we also consider sequences of mappings, but probabilities of answers are computed in a different way. Our aim is to make the probability of an answer dependent *on the number of contexts* in which the answer occurs. For example, $(XML, 2004)$ occurs in two contexts within I_2 (and also in the target instance $J = I_1 \cup I_2 \cup I_3$), i.e. in the context of "Ann" and the context of "$John$". A context will be understood as a subtree (a highest-level subtree) in the target instance J. The subtree is identified by a key value. There are two subtrees in our running example, the *Ann-subtree*, and the *John-subtree*, corresponding to key values "Ann" and "$John$", respectively. The subtrees are ordered in the document order.

Let $(a_1, ..., a_s)$ be a tuple of key values determining subtrees in the target instance J. Let $(m_1, ..., m_s)$ be a sequence of mappings from source instances to J. The probability of the sequence $(m_1, ..., m_s)$ is taken into account while computing probability of an answer ans, if ans is inserted into the a_i*-subtree* by the mapping m_i, for some $1 \leq i \leq s$.

For example, the sequence (m_1, m_3) returns $(XML, 2004)$. However, it is returned by the *second* mapping, i.e. m_3, and inserted into the *first* subtree, i.e. the *Ann-subtree*. Thus, the probability of (m_1, m_3) is not taken into account while computing the probability of $(XML, 2004)$ (see Table 2) according to the *by-subtree* semantics – denoted by N in the *by-subtree* column. However, it is taken into account by the *by-sequence* semantics.

In comparison to the *by-peer* and *by-sequence* semantics, the *by-subtree* better assess trustworthiness of answers. It takes into account the number of contexts in which the answer occurs. For example, since $(XML, 2004)$ occurs in two contexts in the source I_2, the *by-subtree* semantics takes it into account and in the result its probability is higher than this of $(XML, 2005)$.

3 XML Schemas and Instances

In this paper, XML schemas will be specified by *tree-pattern formulas* [3,13]. It means that we will restrict ourselves to a subset of XML schemas – namely, to schemas without recursions and alternatives.

Definition 1. *A schema over a set L of labels conforms to the syntax:*

$$\begin{aligned} S &::= /l[E] \\ E &::= l = x \mid l[E] \mid E \wedge ... \wedge E, \end{aligned} \tag{2}$$

where $l \in L$, and x is a variable. If variable names are significant, we will write $S(\mathbf{x})$, where $\mathbf{x}$ is a vector of text-valued variables.

Example 1. The schema S_3 in Figure 1 has the following specification:
$S_3(x_1, x_2, x_3) := /authors[author[name = x_1 \wedge book[title = x_2 \wedge year = x_3]]]$.

An XML database consists of a set of XML data. We define XML data as an unordered rooted node-labeled tree (XML tree) over a set L of labels, and a set $Str \cup \{\perp\}$ of strings and the distinguished null value $\perp$ (both strings and the null value, $\perp$, are used as values of text nodes).

Definition 2. *An XML tree I is a tuple $(r, N^e, N^t, child, \lambda, \nu)$, where:*

- *r is a distinguished root node, N^e is a finite set of element nodes, and N^t is a finite set of text nodes;*
- *$child \subseteq (\{r\} \cup N^e) \times (N^e \cup N^t)$ – a relation introducing tree structure into the set $\{r\} \cup N^e \cup N^t$, where r is the root, each element node has at least one child (which is an element or text node), text nodes are leaves;*
- *$\lambda : N^e \to L$ – a function labeling element nodes with names (labels);*
- *$\nu : N^t \to Str \cup \{\perp\}$ – a function labeling text nodes with text values from Str or with the null value $\perp$.*

It will be useful to perceive an XML tree I with schema S over variables $\mathbf{x}$, as a pair (S, Ω) (called a *description*), where S is the schema, and Ω is a set of valuations of variables in $\mathbf{x}$. A valuation $\omega \in \Omega$ is a function assigning values from $Str \cup \{\perp\}$ to variables in $\mathbf{x}$, i.e. $\omega : \mathbf{x} \to Str \cup \{\perp\}$.

Example 2. The instance I_3 in Figure 1 can be represented by the following description:
$I_3 := (S_3(x_1, x_2, x_3), \{(Ann, XML, 2004), (Ann, SQL, 2004), (John, C\#, 2005)\})$.

An XML tree I satisfies a description (S, Ω), denoted $I \models (S, \Omega)$, if I satisfies (S, ω) for every $\omega \in \Omega$, where this satisfaction is defined as follows:

Definition 3. *Let S be a schema over $\mathbf{x}$, and ω be a valuation for variables in $\mathbf{x}$. An XML tree I satisfies S by valuation ω, denoted $I \models (S, \omega)$, if the root r of I satisfies S by valuation ω, denoted $(I, r) \models (S, \omega)$, where:*

1. $(I, r) \models (/l[E], \omega)$, *iff* $\exists n \in N^e\ child(r, n) \wedge (I, n) \models (l[E], \omega)$;
2. $(I, n) \models (l[E], \omega)$, *iff* $\lambda(n) = l$ *and* $\exists n' \in N^e(child(n, n') \wedge (I, n') \models (E, \omega))$;
3. $(I, n) \models (l = x, \omega)$, *iff* $\lambda(n) = l$ *and* $\exists n' \in N^t(child(n, n') \wedge \nu(n') = \omega(x))$;
4. $(I, n) \models (E_1 \wedge ... \wedge E_k, \omega)$, *iff* $(I, n) \models (E_1, \omega) \wedge \cdots \wedge (I, n) \models (E_k, \omega)$.

A description (S, Ω) represents a class of S instances with the same set of values (the same Ω), since elements in instance trees can be grouped and nested in different ways. By a *canonical instance* we will understand the instance with the maximal width, i.e. the instance where subtrees corresponding to valuations are pair-wise disjoint. For example, I_1 and I_2 in Figure 1 are canonical, whereas I_3 is not since two books are nested under one author. A canonical instance may be transformed into a required form using specification of keys [13].

4 Schema Mappings

A schema mapping specifies the semantic relationship between a source schema and a target schema. We define it as a *source-to-target dependency* [3,7,13].

Definition 4. *A mapping from a source schema S to a target schema T is an expression of the form*

$$m := \forall \mathbf{x}(S(\mathbf{x}) \Rightarrow \exists \mathbf{y} T(\mathbf{x}', \mathbf{y})), \tag{3}$$

where $\mathbf{x}' \subseteq \mathbf{x}$, and $\mathbf{y} \cap \mathbf{x} = \emptyset$.

A mapping defines one-to-one correspondence between source and target paths. Variable names are used to indicate correspondences between paths bound to variables. In practice, a correspondence also involves a function that transforms values of source and target variables. Using such functions we can express many-to-one and many-to-many correspondences. However, in this paper these functions are irrelevant to our discussion, so they will be omitted.

Example 3. The mapping from S_3 to S_2 is:
$\forall x_1, x_2, x_3(/authors[author[name = x_1 \wedge book[title = x_2 \wedge year = x_3]]] \Rightarrow$
$\exists x_4 / pubs[pub[title = x_2 \wedge year = x_3 \wedge writer[name = x_1 \wedge university = x_4]]])$.

In the following mappings we will omit quantifications.

Example 4. m_1, m_2, and m_3 are mappings from S_1 to S_3, S_2 to S_3, and S_3 to S_3, respectively:

$$\begin{aligned} m_1 :=\ & /books[book[year{=}x_1 \wedge title{=}x_2 \wedge author[name{=}x_3 \wedge university = x_4] \\ & \Rightarrow /authors[author[name = x_3 \wedge book[title = x_2 \wedge year = x_1]]] \\ m_2 :=\ & /pubs[pub[title{=}x_1 \wedge year{=}x_2 \wedge writer[name = x_3 \wedge university = x_4]]] \\ & \Rightarrow /authors[author[name = x_3 \wedge book[title = x_1 \wedge year = x_2]]] \\ m_3 :=\ & /authors[author[name = x_1 \wedge book[title = x_2 \wedge year = x_3]]] \\ & \Rightarrow /authors[author[name = x_1 \wedge book[title = x_2 \wedge year = x_3]]] \end{aligned}$$

A schema mapping m from a source schema S to a target schema T expresses a *constraint*, which is or is not satisfied by a pair (I, J) of XML trees, where I and J are instances of S and T, respectively.

Definition 5. *A pair (I, J) of XML trees satisfies a mapping m, $(I, J) \models m$, if for any valuation ω of variables in $\mathbf{x}$ there is a valuation σ of variables in $\mathbf{y}$ such that:*

$$I \models (S, \omega) \Rightarrow J \models (T, (\omega', \sigma)),$$

where ω' is the restriction of ω to the set $\mathbf{x}'$, denoted $\omega' = \omega[\mathbf{x}']$. Then we say that J is consistent *with I and m.*

In general, there may be zero or many different target instances J consistent with a given source instance I and a given mapping m [7,3].

Definition 6. *An XML data integration setting (XDI) is a triple $(\mathbf{S}, T, \mathbf{M}_{\mathbf{S}T})$, where:*

- $\mathbf{S} = (S_1, ..., S_n)$ – *an ordered set of source schemas;*
- T – *a target schema,*
- $\mathbf{M}_{\mathbf{S}T} = (m_1, ..., m_n)$ – *a set of mappings; m_i is a mapping from S_i to T.*

The following definition specifies the notion of consistency between a target instance J, a set of source instances $(I_1, ..., I_n)$ (called a *complex* source instance $\mathbf{I}$), and a given set of mappings $(m_1, ..., m_n)$.

Definition 7. *Let $\mathbf{M}_{\mathbf{S}T} = (m_1, ..., m_n)$ be a set of mappings, where m_i is a mapping from S_i to T. Let $\mathbf{I} = (I_1, ..., I_n)$ be an instance of $(S_1, ..., S_n)$. We say that a pair $(\mathbf{I}, J)$ satisfies $\mathbf{M}_{\mathbf{S}T}$, denoted $(\mathbf{I}, J) \models \mathbf{M}_{\mathbf{S}T}$, if for each m_i, J is consistent with I_i and m_i. Then J is said to be consistent with $\mathbf{I}$ and $\mathbf{M}_{\mathbf{S}T}$.*

5 Queries and Answers

In this paper we consider queries which return tuples of values (valuations) as opposed to arbitrary trees (like in [3]).

Definition 8. *A query over a target schema $T(\mathbf{x}')$, is an expression of the form*

$$q := \{\mathbf{x} \mid \exists \mathbf{x}'' T(\mathbf{x}')\}, \tag{4}$$

where $\mathbf{x}, \mathbf{x}'' \subseteq \mathbf{x}'$, $\mathbf{x} \cap \mathbf{x}'' = \emptyset$.

Example 5. "Get all pairs (title,year)", can be expressed by the following query over the schema S_3:

$$\{(x_2, x_3) \mid \exists x_1(/authors[author[name = x_1 \wedge book[title = x_2 \wedge year = x_3]]])\}$$

Definition 9. *Let $q(\mathbf{x})$ be a query over a schema $T(\mathbf{x}')$ and J be an instance of T. A valuation ω of $\mathbf{x}$ is an* answer *to q against J, denoted $\omega \in q(J)$, if there is a valuation ω' of $\mathbf{x}'$ such that $J \models (T, \omega')$ and $\omega = \omega'[\mathbf{x}]$, i.e. ω is the restriction of ω' on $\mathbf{x}$.*

Now, let us consider an answer to a query in an XML data integration setting. Such answers in data integration settings are referred to as *certain answers* [11].

Definition 10. *Let $q(\mathbf{x})$ be a query over the target schema T in an XDI $(\mathbf{S}, T, \mathbf{M}_{\mathbf{S}T})$. Let $\mathbf{I}$ be an instance of $\mathbf{S}$. A valuation ω of $\mathbf{x}$ is an* answer *(a* certain answer*) to q against $\mathbf{I}$, if ω is an answer to q against every J, where J is the target instance consistent with $\mathbf{I}$ and $\mathbf{M}_{\mathbf{S}T}$, denoted $\omega \in q(\mathbf{M}_{\mathbf{S}T}(\mathbf{I}))$.*

6 Probabilistic XML Data Integration Setting

Probabilistic schema mapping describes a probability distribution of a set of (ordinary) schema mappings. As we mentioned in Section 2, the probability of a mapping is equal to the probability (modeling the reliability) of the data source (peer) that is the domain of the mapping. In this way we define a *probabilistic XML data integration setting*.

Definition 11. *A probabilistic XML data integration setting ($pXDI$) is a quadruple $(\mathbf{S}, T, \mathbf{M}_{\mathbf{S}T}, Prob)$, where $(\mathbf{S}, T, \mathbf{M}_{\mathbf{S}T})$ is an ordinary XDI, and $Prob$ is a probability function over $\mathbf{M}_{\mathbf{S}T}$ such that for each $m \in \mathbf{M}_{\mathbf{S}T}$:*

- $Prob(m) \in [0, 1]$, *and*
- $\Sigma_{m \in \mathbf{M}_{\mathbf{S}T}} Prob(m) = 1$.

An answer to a query q in a $pXDI$ is a pair (ω, p), where ω is an answer to q in ordinary XDI and p is a probability assigned to ω. The probability models uncertainty about haw reliable is data provided by ω. Three methods can be used to compute this probability: *by-peer* and *by-sequence* semantics, which are based on the *by-table* and *by-sequence* semantics proposed in [6], and *by-subtree* a new semantics proposed in this paper. These semantics were informally discussed in Section 2.

6.1 By-Peer Semantics

Let $(\mathbf{S}, T, \mathbf{M}_{\mathbf{S}T}, Prob)$ be a $pXDI$ and $\mathbf{I}$ be a source instance of $\mathbf{S}$. In the *by-peer* interpretation, all the data from one source has the probability determined by the probability of the mapping defined on this data source. The probability of an answer $\omega \in q(\mathbf{M}_{\mathbf{S}T}(\mathbf{I}))$ is the sum of the probabilities of all mappings producing this answer.

For example, answers in Table 1 are produced by mappings m_1, m_2, and m_3 with probabilities 0.5, 0.2, and 0.3, respectively. The probabilities of answers, i.e. of tuples $(title, year)$, are given in Table 3.

Definition 12. *Let q be a query over T in $(\mathbf{S}, T, \mathbf{M}_{\mathbf{S}T}, Prob)$. Let $\mathbf{I}$ be an instance of $\mathbf{S}$. Let $\mathbf{m}(\omega)$ be the subset of $\mathbf{M}_{\mathbf{S}T}$, such that for each $m \in \mathbf{m}(\omega)$, $\omega \in q(m(\mathbf{I}))$ (if m is a mapping from S_i to T, then $m(\mathbf{I}) = m(I_i)$).*

Let $p = \Sigma_{m \in \mathbf{m}(\omega)} Prob(m)$. Then we say that the pair (ω, p) is a by-peer *answer to q with respect to $\mathbf{I}$ and $(\mathbf{S}, T, \mathbf{M}_{\mathbf{S}T}, Prob)$.*

6.2 By-Sequence Semantics

In contrast to the *by-peer* semantics, where all the mappings were considered in separation, in the *by-sequence* model we will consider sequences of mappings. Thus, if there are n mappings (and also n peers and n data sources), we can analyze sequences with length s of (not necessarily distinct) mappings, $s \geq 1$.

In general, if we have n mappings, then there are n^s sequences of length s. Let $(\mathbf{M}_{\mathbf{S}T}, Prob)$ be a probabilistic mapping. By $\mathbf{seq}^s(\mathbf{M}_{\mathbf{S}T})$ will be denoted the set of all sequences of lengths s created from mappings in $\mathbf{M}_{\mathbf{S}T}$. Then we can think of every sequence $seq \in \mathbf{seq}^s(\mathbf{M}_{\mathbf{S}T})$ as a separate event. Probabilities assigned to sequences satisfy the following formulas:

$$Prob(seq) = \Pi_{m \in seq} Prob(m),$$
$$\Sigma_{seq \in \mathbf{seq}^s(\mathbf{M}_{\mathbf{S}T})} Prob(seq) = 1.$$

Each sequence $seq \in \mathbf{seq}^s(\mathbf{M}_{\mathbf{S}T})$ creates an instance of the target schema T. This instance will be denoted by $J_{seq} = seq(\mathbf{I}) = \bigcup_{m \in seq} m(\mathbf{I})$ and it is consistent with $\mathbf{I}$ and seq, i.e.:

- for each $m_k \in seq$, J_{seq} is consistent with I_k and m_k,
- for each ω, if $J_{seq} \models (T, \omega)$ then there is $m_k \in seq$ such that $I_k \models (S_k, \omega)$.

In our running example, we consider sequences of length 2. There are 9 such sequences (see Table 2). According to the above considerations, an answer in the *by-sequence* semantics is defined as follows:

Definition 13. *Let q be a query over T in $(\mathbf{S}, T, \mathbf{M}_{\mathbf{S}T}, Prob)$. Let $\mathbf{I}$ be an instance of $\mathbf{S}$. Let $\mathbf{seq}(\omega)$ be the subset of $\mathbf{seq}^s(\mathbf{M}_{\mathbf{S}T})$, such that for each $seq \in \mathbf{seq}(\omega)$, $\omega \in q(seq(\mathbf{I}))$.*

Let $p = \Sigma_{seq \in \mathbf{seq}(\omega)} Prob(seq)$. Then we say that the pair (ω, p) is a by-sequence *answer to q with respect to $\mathbf{I}$ and $(\mathbf{S}, T, \mathbf{M}_{\mathbf{S}T}, Prob)$.*

In Table 3 there are also probabilistic answers according to the *by-sequence* semantics in our running example.

6.3 By-Subtree Semantics

While computing a probabilistic answer in the *by-subtree* semantics we take into account that the answer may occur in many *contexts* in the target instance. The

more contexts in which the answer occurs the highest is the probability of the answer (i.e. the answer is more likely to be correct). Note that the *by-peer* and *by-sequence* semantics are not sensitive to the number of contexts containing the considered answer.

As the contexts we assume the highest-level subtrees in the target instance. The subtree is identified by an absolute XML key [5] or a key functional dependency [1,13]. For example, for the schema S_3 we can define the key functional dependence $/authors/author/name \rightarrow /authors/author$. In our approach this key functional dependency can be equivalently expressed by the following path formula [13]:

$$\kappa(x_1) := /authors/author[name = x_1] \tag{5}$$

meaning that for a given value of x_1, the value of $\kappa(x_1)$ contains at most one node (the root of the subtree of type $/authors/author$ determined by x_1). Then an instance of S_3 contains as many subtrees, of type $/authors/author$, as there are different text values of the path $/authors/author/name$.

In our running example we have two subtrees determined by "Ann" and "$John$", respectively. Thus, we can consider sequences of length 2 of mappings as in the *by-sequence* semantics. However, we will use another semantics to compute probabilities of answers (as was informally discussed in Section 2).

Similarly as in *by-sequence* semantics, let q be a query over a target schema T, $\mathbf{I} = (I_1, ..., I_n)$ be an instance of $(S_1, ..., S_n)$, and $(\mathbf{M_{S\,T}}, Prob)$ be a probabilistic mapping. Let J be an instance of T with s subtrees, and let J be by-sequence consistent with $\mathbf{I}$ and $\mathbf{M_{S\,T}}$. Let $\kappa(\mathbf{z})$ be the key definition over S, where $\mathbf{z}$ is a vector of text variables. Then there are s different values of $\mathbf{z}$, say $\mathbf{a_1}$,..., $\mathbf{a_s}$.

In the *by-subtree* semantics, probabilistic answers of q are defined as follows:

Definition 14. *Let* $\mathbf{subtree}(\omega)$ *be the subset of* $\mathbf{seq}^s(\mathbf{M_{S\,T}})$*, such that for each* $seq = (m^1, ..., m^s) \in \mathbf{subtree}(\omega)$

$$\omega \in q[\mathbf{z} \mapsto \mathbf{a_1}](m^1(\mathbf{I}))) \cup \cdots \cup q[\mathbf{z} \mapsto \mathbf{a_s}](m^s(\mathbf{I}))),$$

where $q[\mathbf{z} \mapsto \mathbf{a_i}]$ *is a query created from* q *by substituting variables in* $\mathbf{z}$ *(the key variables) by text values from the vector* $\mathbf{a_i}$*.*

Let $p = \Sigma_{seq \in \mathbf{subtree}(\omega)} Prob(seq)$*. Then we say that* ω *is a* by-sebtree *answer to* q *with probability* p*, with respect to* $\mathbf{I}$ *and* $(\mathbf{M_{S\,T}}, Prob)$*.*

Example 6. For the sequence (m_1, m_2) of mappings (Example 4), the query q (Example 5), and the key (5), we have (compare Table 2):

$$q[x_1 \mapsto "Ann"](m_1(I_1)) = \{(C\#, 2006)\},$$
$$q[x_1 \mapsto "John"](m_2(I_2)) = \{(XML, 2004)\}.$$

7 Conclusion

In this paper we discussed an approach to probabilistic XML data integration systems. We use probabilities to model reliabilities of data sources and use them

to compute probabilistic answers. We discuss three ways to determine probabilistic answers: *by-peer*, *by-sequence*, and *by-subtree* semantics. We claim that the *by-subtree* semantics has the advantage over two others, since it more precisely computes probabilistic answers. This is significant contribution of this paper. Probabilities associated to inconsistent answers can be used to select these which are more likely to be correct and can be used to resolve inconsistencies violating XML functional dependencies.

Acknowledgement. The work was supported in part by the Polish Ministry of Science and Higher Education under Grant N516 015 31/1553.

References

1. Arenas, M.: Normalization theory for XML. SIGMOD Record 35(4), 57–64 (2006)
2. Arenas, M., Bertossi, L.E., Chomicki, J.: Consistent Query Answers in Inconsistent Databases. In: PODS, pp. 68–79 (1999)
3. Arenas, M., Libkin, L.: XML Data Exchange: Consistency and Query Answering. In: PODS Conference, pp. 13–24 (2005)
4. Bohannon, P., Flaster, M., Fan, W., Rastogi, R.: A Cost-Based Model and Effective Heuristic for Repairing Constraints by Value Modification. In: SIGMOD Conference, pp. 143–154 (2005)
5. Buneman, P., Davidson, S.B., Fan, W., Hara, C.S., Tan, W.C.: Reasoning about keys for XML. Information Systems 28(8), 1037–1063 (2003)
6. Dong, X.L., Halevy, A.Y., Yu, C.: Data Integration with Uncertainty. In: VLDB, pp. 687–698. ACM, New York (2007)
7. Fagin, R., Kolaitis, P.G., Popa, L., Tan, W.C.: Composing Schema Mappings: Second-Order Dependencies to the Rescue. In: PODS, pp. 83–94 (2004)
8. Fuxman, A., Fazli, E., Miller, R.J.: ConQuer: Efficient Management of Inconsistent Databases. In: SIGMOD Conference, pp. 155–166 (2005)
9. Greco, G., Lembo, D.: Data Integration with Preferences Among Sources. In: Atzeni, P., Chu, W., Lu, H., Zhou, S., Ling, T.-W. (eds.) ER 2004. LNCS, vol. 3288, pp. 231–244. Springer, Heidelberg (2004)
10. Greco, S., Sirangelo, C., Trubitsyna, I., Zumpano, E.: Preferred Repairs for Inconsistent Databases. In: IDEAS 2003, pp. 202–211. IEEE Computer Society, Los Alamitos (2003)
11. Lenzerini, M.: Data Integration: A Theoretical Perspective. In: Popa, L. (ed.) PODS, pp. 233–246. ACM, New York (2002)
12. Madhavan, J., Halevy, A.Y.: Composing Mappings Among Data Sources. In: VLDB, pp. 572–583 (2003)
13. Pankowski, T.: XML data integration in SixP2P – a theoretical framework. In: EDBT 2008 Workshop on Data Management in P2P Systems, ACM Digital Library (2008)
14. Rahm, E., Do, H.H.: Data Cleaning: Problems and Current Approaches. IEEE Data Eng. Bull. 23(4), 3–13 (2000)
15. Staworko, S., Chomicki, J., Marcinkowski, J.: Preference-Driven Querying of Inconsistent Relational Databases. In: Grust, T., Höpfner, H., Illarramendi, A., Jablonski, S., Mesiti, M., Müller, S., Patranjan, P.-L., Sattler, K.-U., Spiliopoulou, M., Wijsen, J. (eds.) EDBT 2006. LNCS, vol. 4254, pp. 318–335. Springer, Heidelberg (2006)
16. Taylor, N.E., Ives, Z.G.: Reconciling while tolerating disagreement in collaborative data sharing. In: SIGMOD Conference, pp. 13–24. ACM, New York (2006)