A probabilistic XML approach to data integration

Maurice van Keulen
m.vankeulen@utwente.nl

Ander de Keijzer
a.dekeijzer@utwente.nl

Wouter Alink
w.alink@ewi.utwente.nl

Faculty of EEMCS, University of Twente
POBox 217, 7500 AE Enschede, The Netherlands

Abstract

In mobile and ambient environments, devices need to become autonomous, managing and resolving problems without interference from a user. The database of a (mobile) device can be seen as its knowledge about objects in the 'real world'. Data exchange between small and/or large computing devices can be used to supplement and update this knowledge whenever a connection gets established. In many situations, however, data from different data sources referring to the same real world objects, may conflict. It is the task of the data management system of the device to resolve such conflicts without interference from a user. In this paper, we take a first step in the development of a probabilistic XML DBMS. The main idea is to drop the assumption that data in the database should be certain: subtrees in XML documents may denote possible views on the real world. We formally define the notion of probabilistic XML tree and several operations thereon. We also present an approach for determining a logical semantics for queries on probabilistic XML data. Finally, we introduce an approach for XML data integration where conflicts are resolved by the introduction of possibilities in the database.

1 Introduction

In mobile and ambient environments, devices need to become autonomous, managing and resolving problems without interference from a user. This also holds for the data management component in the device. The database of a (mobile) device can be seen as its knowledge about objects in the 'real world'. An XML fragment $\langle\mathsf{person}\rangle\langle\mathsf{nm}\rangle\mathrm{John}\langle/\mathsf{nm}\rangle\langle\mathsf{tel}\rangle 1111\langle/\mathsf{tel}\rangle\langle/\mathsf{person}\rangle$ represents knowledge about a person in the real world whose name is "John" and whose telephone number is "1111".

Data exchange between small and/or large computing devices can be used to supplement and update this knowledge whenever a network connection gets established. In many situations, however, data from different data sources referring to the same real world objects, may conflict. Interaction with the user to resolve the conflict is not an option. Imagine your mobile phone beeping every time you get into Bluetooth range of a friendly device that sends you information that partially conflicts with your own. It is the task of the data management system of the device to resolve such conflicts without interference from a user.

In this paper, we take a first step in the development of a probabilistic XML DBMS capable of supporting data integration. The main idea is to drop the assumption that data in the database should be certain: subtrees in XML documents may denote possible views on the real world. For example, the subtree for our person "John" may contain uncertainty on his telephone number by enumerating the possibilities that he either has phone number "1111" or "2222". In this way, attempting to integrate data from data sources that conflicts on the phone number of "John", the data management component need not consult the user, but simply stores his own uncertainty in the database. Obviously, other data management components and applications using them need to support uncertainty in data too. For example, a query for the phone number of persons named "John" results in an uncertain answer: it is either "1111" or "2222".

Although this paper focuses on data integration, there are other application areas where probabilistic (XML) databases are useful. For example, in computer vision, image retrieval and document annotation, objects/classes need to be identified based on text, image or video content. This is an error-prone process, so the resulting (meta-)data is uncertain. [11] specifically mentions information extraction and scientific data management as application areas for probabilistic databases. In [6], probabilistic data is also used for the purpose of data integration, but from an information retrieval point of view. They construct a probability tree indicating the level of certainty with which a document is about a certain topic, or field.

In this first attempt at a probabilistic XML DBMS capable of managing uncertainty in its data resulting from data

Proceedings of the 21st International Conference on Data Engineering (ICDE 2005)
1084-4627/05 $20.00 © 2005 **IEEE**

integration, we make a few *simplifying assumptions*. We believe, that under these assumptions, the system is simple enough to study the fundamental problems, while it can still manage uncertainty as a result of data integration adequately.

- The model defined in Section 3 is rather simple. It is based on enumerating possibilities. More advanced models are known [8, 9], but this simple one appears to be adequate for a first attempt at data integration.
- As customary for database, the device assumes for querying that its database contains all knowledge (closed world assumption). Only at data integration time, it acknowledges that other devices may know more.
- Other devices are assumed to deliver information according to the proposed approach in an honest and predictable way.
- Schemas of documents are available. For now, we also assume that schemas of to be integrated documents are the same, so we can concentrate on data integration rather than schema integration.

Roughly, we propose the following *strategy* for data integration. The information in a foreign probabilistic XML document is used to supplement a device's knowledge of the real world. We either find (1) information on previously unknown real world objects, (2) conflicting information on already known real world objects, or (3) the same information on already known real world objects. In the first case, that information is simply added. In the second case, we incorporate existing and new information as distinct possibilities. The last case only confirms what the device already knows. Note that the fact that information corresponds to the same real world object can often not be determined with certainty. We add possibilities to the document accordingly. We assume the existence of a *rule engine* that determines the probability of two elements referring to the same real world object.

The paper is organized as follows. We first provide some related research for a large part in the realm of probabilistic *relational* databases. In Section 3, we formally define the notion of probabilistic XML tree and several operations thereon. Section 4 presents an approach for determining a proper semantics for queries on probabilistic XML data. The approach for XML data integration is further detailed in Section 5. In Section 6, we validate our ideas with some experiments. This first attempt at a probabilistic XML DBMS gave rise to many issues that need to be resolved. We list them in Section 7.

2 Related research

Our model for representing uncertainty is based on experience from probabilistic *relational* databases. A central notion in the relational model is the tuple. Probabilistic relational databases often associate probabilities with tuples to indicate the level of confidence in the existence of the entire tuple. Probabilities are also often associated with individual attributes to indicate the level of confidence in the value for that particular attribute. [12] defines these as Type-1 and Type-2 probabilities, respectively. In XML, probabilities are associated with elements. This probability, therefore, indicates the level of confidence in that element, but, at the same time, also in the existence of the subtree rooted in that element. As a result, the distinction between Type-1 and Type-2 disappears in XML.

In [2, 1], dependencies among probabilistic attributes are not allowed. This is, among others, due to the fact that dependencies are difficult to express properly in the relational model. As [11] also discovered, dependencies like mutual exclusiveness and simultaneous occurrence can be expressed in a natural way in XML.

In [2], we proposed a probabilistic relational model supporting Type-2 probabilities. The possible world approach in this case, required exactly one possibility for each real world object to be present in every possible world, in other words, it could not express uncertainty about (non-)existence of an object. This requirement disappears in the case of probabilistic XML, i.e., we are able to support uncertainty about existence of real world objects.

In [8, 9], a model for uncertain semistructured data is proposed. Their approach starts from a graph theoretical point of view and adds support and constraints to the model to facilitate uncertain information. Unlike our approach, they associate probabilities with individual nodes, whereas we model possibilities as separate nodes in a tree.

3 Formalization of Probabilistic XML

Since order is important in XML, we first introduce some notation for handling sequences.

Notational convention 1 *Analogous to the powerset notation $\mathbb{P}$, we use a* power sequence *notation $\mathbb{S}\,A$ to denote the domain of all possible sequences built up of elements of A. We use the notation $[a_1, \ldots, a_n]$ for a particular sequence. We use set operations for sequences, such as $\cup, \exists, \in$, whenever definitions remain unambiguous.*

We start by defining the notions of *tree* and *subtree* as abstractions of an XML document and fragment. We model a tree as a node and a sequence of child subtrees. We abstract from the details of nodes. We assume that they incorporate *properties* like tag name, node identity, node kind, and, if appropriate, attribute or text value. Equality on nodes is defined as equality on the properties.

Definition 2 *Let $\mathcal{N}$ be the set of* nodes*. Let $\mathcal{T}_i$ be the set of* trees *with maximum level i inductively defined as follows:*

$$\begin{aligned}\mathcal{T}_0 &= \{(n, \emptyset) \mid n \in \mathcal{N}\}\\ \mathcal{T}_{i+1} &= \mathcal{T}_i \cup \{(n, ST) \mid n \in \mathcal{N} \\ &\qquad \wedge ST \in \mathbb{S}\,\mathcal{T}_i \\ &\qquad \wedge (\forall T \in ST \bullet n \notin \mathcal{N}^T) \\ &\qquad \wedge (\forall T, T' \in ST \bullet T \neq T' \\ &\qquad\quad \Rightarrow \mathcal{N}^T \cap \mathcal{N}^{T'} = \emptyset)\}\end{aligned}$$

where $\mathcal{N}^T = \{n\} \cup \bigcup_{T' \in ST} \mathcal{N}^{T'}$. Let $\mathcal{T}_{\text{fin}}$ be the set of finite trees*, i.e., $T \in \mathcal{T}_{\text{fin}} \Leftrightarrow \exists i \in \mathbb{N} \bullet T \in \mathcal{T}_i$. In the sequel, we only work with finite trees. We sometimes use $\overline{n}$ to indicate the* root node *of a tree.*

We obtain a subtree from a tree T by indicating a node n in T which is the root node of the desired subtree. We also define a convenience function child that returns the child nodes of a given node in a tree.

Definition 3 *Let* $\mathsf{subtree}(T, n)$ *be the* subtree *within $T = (\overline{n}, ST)$ rooted at n.*

- $\mathsf{subtree}(T, n) = \begin{cases} T & \text{if } \overline{n} = n \\ \mathsf{subtree}(T', n) & \text{otherwise} \end{cases}$

where T' such that $(T', ST') \in ST \wedge n \in \mathcal{N}^{T'}$. For $\mathsf{subtree}(T, n) = (n, [(n_1, ST_1), \ldots, (n_m, ST_m)])$, let $\mathsf{child}(T, n) = [n_1, \ldots, n_m]$.

The central notion in our paper is the *probabilistic tree*. In an ordinary XML document, all information is certain. When two XML data sources are integrated, they may conflict on information about certain real world objects. Therefore, after data integration, there may exist more than one possibility for a certain text node, or in general, for entire subtrees. We model this uncertainty in a probabilistic tree by introducing two special kinds of nodes:

1. probability nodes depicted as $\bigtriangledown$, and
2. possibility nodes depicted as $\circ$, which have an associated probability.

The children of a probability node enumerate all possibilities with a combined probability of 1. Ordinary XML nodes are depicted as $\bullet$. A probabilistic tree is well-structured, if the children of a probability node are possibility nodes, the children of a possibility node are XML nodes, and the children of XML nodes are probability nodes. In this way, on each level of the tree, you only find one kind of nodes.

Figure 1 shows an example of a probabilistic XML tree. The tree represents an XML document with a root node 'persons' (which exists with certainty). The root node has either one or two child nodes 'person' (with probabilities .7 and .3, respectively). In the one-child case, the name of the person is 'John' and the telephone number is either

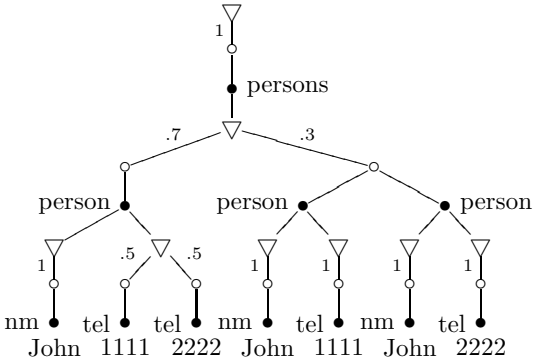


Figure 1. Example probabilistic XML tree.

'1111' or '2222' (with equal probability). In the more unlikely case of two children, the information of both persons is certain, i.e., they both have name 'John' and one has telephone number '1111' and the other '2222'.

This document is a possible result of two documents having been integrated, one document stating the telephone number of a person named 'John' to be '1111', and the other stating the telephone number of a person named 'John' to be '2222'. It is uncertain if both talk about the same person. A data integration matching rule apparently determined that, with a probability of .7, they represent the same person. Therefore, the combined knowledge of the real world is described accurately by the given tree.

A probabilistic tree is defined as a tree, a kind function that assigns node kinds, and a prob function that assigns probabilities to possibility nodes. The root node is defined to always be a probability node. A special type of probabilistic tree is a *certain* one, which means that all information in it is certain, i.e., there is no more than one possibility for any node.

Definition 4 *A* probabilistic tree $PT = (T, \mathsf{kind}, \mathsf{prob})$ *is defined as follows*

- $\mathsf{kind} \in \mathcal{N} \to \{\underline{\mathsf{prob}}, \underline{\mathsf{poss}}, \underline{\mathsf{xml}}\}$
- $\mathcal{N}^T_k = \{n \in \mathcal{N}^T \mid \mathsf{kind}(n) = k\}$.
- $\mathsf{kind}(\overline{n}) = \underline{\mathsf{prob}}$ *where* $T = (\overline{n}, ST)$
- $\forall n \in \mathcal{N}^T_{\underline{\mathsf{prob}}} \forall n' \in \mathsf{child}(T, n) \bullet n' \in \mathcal{N}^T_{\underline{\mathsf{poss}}}$
- $\forall n \in \mathcal{N}^T_{\underline{\mathsf{poss}}} \forall n' \in \mathsf{child}(T, n) \bullet n' \in \mathcal{N}^T_{\underline{\mathsf{xml}}}$
- $\forall n \in \mathcal{N}^T_{\underline{\mathsf{xml}}} \forall n' \in \mathsf{child}(T, n) \bullet n' \in \mathcal{N}^T_{\underline{\mathsf{prob}}}$
- $\mathsf{prob} \in \mathcal{N}^T_{\underline{\mathsf{poss}}} \to [0, 1]$
- $\forall n \in \mathcal{N}^T_{\underline{\mathsf{prob}}} \bullet (\sum_{n' \in \mathsf{child}(T,n)} \mathsf{prob}(n')) = 1$.

A probabilistic tree $PT = (T, \mathsf{kind}, \mathsf{prob})$ is certain *iff there is only one possibility node for each probability node, i.e.,* $\mathsf{certain}(PT) \Leftrightarrow \forall n \in \mathcal{N}^T_{\underline{\mathsf{prob}}} \bullet |\mathsf{child}(T, n)| = 1$. *To clarify definitions, we use b to denote a probability node, s to denote a possibility node, and x to denote an XML node.*

Subtrees under probability nodes denote *local* possibilities. In the one-person case of Figure 1, there are two local possibilities for the phone number, it is either '1111' or '2222'. The other uncertainty in the tree are the possibilities that there are one or two persons. Viewed globally and from the perspective of a device with this data in its database, the real world could either look like

- one person with name 'John' and phone number '1111' (probability $.5 \times .7 = .35$),
- one person with name 'John' and phone number '2222' (probability $.5 \times .7 = .35$), or
- two persons with name 'John' and respective phone numbers '1111' and '2222' (probability .3).

These are called *possible worlds*.

Definition 5 *A certain probabilistic tree PT' is a* possible world *of another probabilistic tree PT, i.e.,* $\mathsf{pw}(PT', PT)$*, with* probability $\mathsf{pwprob}(PT', PT)$ *iff*

- $PT = (T, \mathsf{kind}, \mathsf{prob}) \wedge PT' = (T', \mathsf{kind}', \mathsf{prob}')$
- $T = (\overline{n}, ST_{\overline{n}}) \wedge T' = (\overline{n}, ST'_{\overline{n}})$
- $\exists s \in \mathsf{child}(T, \overline{n}) \bullet \mathsf{child}(T', \overline{n}) = [s]$
- $X = \mathsf{child}(T, s) = \mathsf{child}(T', s)$
- $\forall x \in X \bullet \mathsf{child}(T, x) = \mathsf{child}(T', x)$
- $B = \bigcup_{x \in X} \mathsf{child}(T, x)$
- $\forall b \in B \bullet PT_b = \mathsf{subtree}(PT, b) \wedge PT'_b = \mathsf{subtree}(PT', b) \wedge \mathsf{pw}(PT'_b, PT_b)$
- $\forall b \in B \bullet p_b = \mathsf{pwprob}(PT_b, PT'_b)$
- $\mathsf{pwprob}(PT', PT) = \mathsf{prob}(s) \times \prod_{b \in B} p_b$

The set of all possible worlds of a probabilistic tree PT is $\mathsf{PWS}_{PT} = \{PT' \mid \mathsf{pw}(PT', PT)\}$.

A probabilistic tree is a compact representation of the set of all possible worlds, but there is more than one possible representation. The optimal representation is the one with the least number of nodes obtained through a process called simplification.

Definition 6 *Two probabilistic trees PT_1 and PT_2 are* equivalent *iff* $\mathsf{PWS}_{PT_1} = \mathsf{PWS}_{PT_2}$. *$PT_1$ is* more compact *than PT_2 if $\left|\mathcal{N}^{PT_1}\right| < \left|\mathcal{N}^{PT_2}\right|$. The transformation of a probabilistic tree to an equivalent more compact one is called* simplification.

Figure 2 shows an example of two equivalent probabilistic trees. They both denote the set of possible worlds containing trees with

- two nodes 'nm' and 'tel' with child text nodes 'John' and '1111' respectively (probability .8) and
- two nodes 'nm' and 'tel' with child text nodes 'John' and '2222' respectively (probability .2).

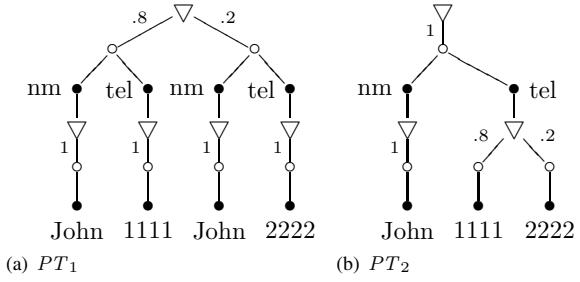


Figure 2. Probabilistic XML tree equivalence.

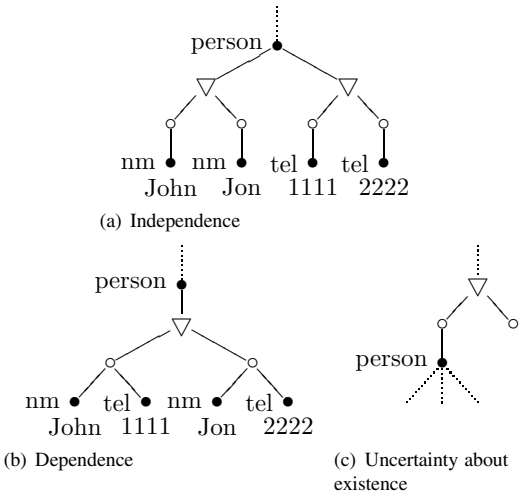


Figure 3. Expressiveness of probabilistic tree model.

As mentioned earlier, relational approaches often disallow dependencies among attributes. The higher expressiveness of the probabilistic tree makes such a restriction unnecessary. Figure 3 illustrates three common patterns: independence between attributes (Figure 3(a)): any combination of 'nm' and 'tel' is possible, dependency between attributes (Figure 3(b)): only the combinations 'John'/'1111' and 'Jon'/'2222' are possible, and uncertainty about the existence of an object (Figure 3(c)): one possibility is empty, i.e., has no subtree. These patterns can occur on any level in the tree, which allows a much larger range of situations to be expressed.

4 Query Evaluation

4.1 Approach

For relational probilistic databases, several extensions to relational algebra to support uncertain data have been proposed [1, 10, 3, 7, 5]. Based on [12], we argued in [2] that thinking in terms of possible worlds is powerful in determining a proper semantics of queries. Uncertainty can be treated as having more than one possible instantiation describing a particular real world object. Choosing one possible instantiation, or *possibility* for short, for each real world object, results in a possible world. Analogous to the notion of parallel universes, all possible worlds co-exist in the database and a query should, therefore, be evaluated in every possible world separately. This approach is not specific to relational databases, so we have adopted it for probabilistic XML as well.

The parallel universes analogy also illustrates that a query on uncertain data may produce a set of possible answers indicating that the answer is uncertain as well. One obtains one possible answer per possible world. The probability of a possible answer is simply the probability of the possible world that gave rise to it.

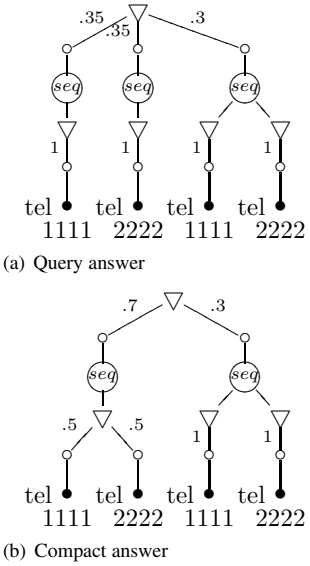
In short, the correct answer to a query on a probabilistic XML document can be obtained as follows:

1. enumerate all n possible worlds for the probabilistic tree,
2. evaluate the query for each possible world according to ordinary semantics,
3. the resulting n possible answers can be simplified by merging (partially) equal possible answers.

This approach applies to queries in any query language. In XPath or XQuery, answers to queries are sequences. Queries on uncertain data, hence, result in sets of possible sequences. Assuming that a sequence is represented by a special sequence node $\textcircled{seq}$, a probabilistic tree can be used to represent the set of possible answers and simplification can be used to obtain a compact representation of the answer.

Note that this is just a statement about the semantics of a query. An implementation can obviously use a more efficient algorithm for answering queries than by enumerating all possible worlds. Concerning query evaluation, our prototype implementation focuses on obtaining insight into how and to what degree possibilities in the data propagate to possibilities in the answer. It, therefore, simply follows the three-step strategy presented above. Performance of query evaluation is a topic for future research and, hence, beyond the scope of this paper. We only sketch how the prototype enumerates all possible worlds in Subsection 4.2 below.

Figure 4 shows an answer to an example query on the data of Figure 1. We have seen that there are three possible worlds. Evaluating the query in each of these possible worlds gives rise to the answer in Figure 4(a). Intuitively, it states that the answer to telephone numbers of people named "John" is uncertain: it is either "1111" or "2222" or there are actually two Johns with a telephone number: one with "1111" and one with "2222". Figure 4(b) shows how we can compactly represent this answer by merging the first to possibilities, i.e., making the possibilities local, since in both cases it is certain that the sequence contains one element.



(a) Query answer

(b) Compact answer

Figure 4. Query `//person[nm="John"]/tel`

Note that to applications, it may be more logical to present the answer to a query as a set of possible answers where each answer is an ordinary XML sequence with ordinary non-probabilistic XML fragments as elements. Such a representation coincides with an enumeration of the possible worlds of the answer. On the other hand, an application prepared for consuming probabilistic answers may benefit in performance by the compact representation. The design of an API is still an open issue.

4.2 Enumerating possible worlds

Generating all possible worlds from a compact probabilistic XML tree can be done recursively. The approach of our prototype implementation is as follows. Given a node in the tree, the function produces all possible worlds for the subtree rooted at that node. Contrary to the formalization, a possible world is represented by a normal XML tree instead of a certain probabilistic tree, i.e., it contains only XML

nodes and no probability and possibility nodes.

For producing the set of possible worlds for a certain node, the sets of possible worlds from its children need to be combined. There are two ways to combine sets of possible worlds:

- The *product* of two sets of possible worlds $W_1 = \{A_1, \ldots, A_n\}$ and $W_2 = \{B_1, \ldots, B_m\}$ is $W_1 \otimes W_2 = \{A_1B_1, A_1B_2, \ldots A_1B_m, A_2B_1, A_2B_2, \ldots A_nB_m\}$, where AB means a world in which both A and B hold. For example, A and B may correspond to two distinct children of a certain element, hence, a possible world combines one possibility for each child.
- The *sum* of two sets of possible worlds $W_1 \oplus W_2 = \{A_1, \ldots, A_n, B_1, \ldots, B_m\}$.

Let $W_1, \ldots, W_n$ be the sets of possible worlds for each child of a given node. The resulting set of possible worlds for that node depends on the node kind:

- *Probability node*. The result is $W_1 \oplus \ldots \oplus W_n$.
- *Possibility node*. The result is $W_1 \otimes \ldots \otimes W_n$.
- *XML node*. The result is constructed by placing the given XML node as parent (i.e., root) in each world in $W_1 \otimes \ldots \otimes W_n$.

5 Data Integration

5.1 General approach

A device's database is a probabilistic XML document. When data integration with a foreign probabilistic XML document is initiated, the foreign document is considered to be a source of 'new' information on real world objects the device either already knows about or not. New information on 'new' real world objects is simply added to the database. Any differences in information on 'existing' real world objects are regarded as different possibilities for that object. Note that we disregard possibilities concerning order. New information on 'new' real world objects is simply considered to come after information on known objects in document order.

Since it is often not possible to determine with certainty that two specific XML elements correspond to the same real world object, we assume the existence of a *rule engine* that determines the probability of two elements referring to the same real world object. In special cases, this rule engine may obviously decide on a probability of 0 (with certainty *not* the same real world object) or 1 (with certainty the same real world object). In this paper, we abstract from the details of this rule engine, but imagine that it uses schema information to rule out possibilities. Or it may, for example, consult a digital street map to declare a certain street name very improbable as there exists no such street in that city. Or it may use Semantic Web techniques to reason away possibilities.

In our current prototype, the rule engine is very simple and uses only some basic schema information. It distinguishes two cases:

- The schema states that a certain element can appear only once. We assume that this means that the elements of both documents refer to the same real world object, hence, the subtrees are correspondingly merged. For example, if two person elements refer to the same real world person, their descendant elements (e.g., nm and tel) are merged. If two corresponding descendant elements differ, we store this as two possibilities for that element.
- The schema states that a certain element can appear multiple times. We assume that this means that the foreign document may contain new elements for this list. For example, the database contains knowledge about two persons "John" and "Rita". The foreign document holds information on a person "Jon". Note that "Jon" may be the same person as "John" only misspelled, or it may refer to a different person. The data integrator will store both possibilities, i.e., one whereby it merges "John" and "Jon", and one whereby it adds a new person element.

Each possibility is assigned a probability by the rule engine. For example, it is not unthinkable that "Jon" and "John" are actually the same person. On the other hand, it is rather improbable that "Rita" and "Jon" are the same person. Our current prototype is not that clever though, i.e., it assigns the same probability to the likelihood that two person elements refer to the same real-life person as to the likelihood of both referring to different real-life persons. The reason for this is that we need it as a base line to determine the effectiveness of a more clever rule engine. Moreover, we would like to focus on the data integration mechanism first. The minimal set of rules used by our prototype also includes that there can only be one root in an XML document and schema's of integrated documents are the same, so different tag names are assumed to refer to different real world objects.

5.2 Integrating sequences

In general, integrating sequences produces possibilities for all elements referring to either the same or different real world objects. Since we made an assumption that the schemas are the same and that elements with different tag names refer to different real world objects, many of those possibilities are ruled out. However, this rule does not limit the possibilities for sequences of elements with the same tag name. Take for example, the integration of address information of people. We are confronted with integrating sequences of person elements. Because we initially chose for a rather dumb rule engine, any two elements, one from

Referral to real world object	resulting sequence
$A \neq B \neq C \neq D$	A, B, C, D
$A = C, B \neq C \neq D$	$A/C, B, D$
$A = D, B \neq C \neq D$	$A/D, B, C$
$A \neq C \neq D, B = C$	$A, B/C, D$
$A \neq C \neq D, B = D$	$A, B/D, C$
$A = C, B = D$	$A/C, B/D$
$A = D, B = C$	$A/D, B/C$

Table 1. Possibilities for merging sequences $x = \{A, B\}$ and $y = \{C, D\}$

each sequence, possibly refer to the same real world object. Therefore, when merging two sequences, X and Y, the resulting number of possibilities can be huge.

Let, for example, $X = [A, B]$ and $Y = [C, D]$. The possibilities to be generated during integration of X and Y are listed in Table 1. In the table, $A = C$ indicates that A and C are considered to refer to the same real world object, hence, they should result in a single possibility where A and B are merged: A/B. Since the database already represents all possibilities explicitly, we do not need to consider two elements from one sequence to refer to the same real world object, so $A = B$ and $C = D$ are not valid possibilities.

When all elements of X and Y refer to other real world objects, the number of resulting possible worlds is 1. But, when one element from X refers to the same real world object as an element from Y, there are $X \times Y$ possible ways how this can be done, since every element from X can in principle be matched with every element from Y.

In general, the number of possible ways to merge i elements from X with i elements from Y can be computed as follows. Choose i different elements from X, where the order of choosing the elements is unimportant, but an element cannot be chosen more than once. This can be done in $\frac{x!}{(x-i)!i!} = \binom{x}{i}$ ways. Then, we choose i elements from Y to merge with those chosen from X. Since the first chosen element from X should be merged with the first element of Y, order is important when choosing elements from Y. The number of ways to choose the i elements from Y is $\frac{y!}{(y-i)!}$.

Let x and y be the lengths of X and Y, respectively. The process of merging sequences is commutative, we assume $x \leq y$. In determining all possibilities, any i ($0 \leq i \leq x$) elements of X may refer to the same real world object as elements of Y. Therefore, the resulting total number of possibilities for a merged sequence is

$$\sum_{i=0}^{x} \binom{x}{i} \times \frac{y!}{(y-i)!}$$

We see from this formula, that merged documents can become huge quite rapidly. If we take, for example, $x = 5$ and $y = 5$, then the maximum number of possibilities is 1546. The rule engine, however, may rule out certain possibilities. For example, if in the case of Table 1, A refers to a person named "John" and C to a person named "Rita", the rule engine may assign probability 0 to the likelihood that $A = C$. In this way, it rules out two of the seven possibilities.

5.3 Data integration is an equivalence preserving operation

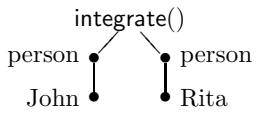
An interesting property of the data integration approach described above, is if it preserves equivalence. Let D_1 and D_2 be two probabilistic XML documents and $A = \mathsf{integrate}(D_1, D_2)$ the result of integrating them. Suppose D_1' and D_2' are equivalent to D_1 and D_2 respectively. Is then $A' = \mathsf{integrate}(D_1', D_2')$ equivalent to A? A full proof goes beyond the scope of this paper, but below we try to show that it is plausible.

There is a special case for which this property is especially interesting. The set of possible worlds can be represented as a probabilistic tree with one probabilistic node as root and all possible worlds as possibilities directly below it. Figure 2(a) is of this form. Since the above property holds, integrating two probabilistic trees amounts to integrating all combinations of possible worlds of both trees.

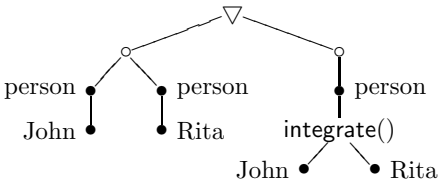
Below, we first show an example of integrating two certain trees to illustrate the recursive process. We then show the integration of a compact tree with two possibilities with a certain tree. Finally, we show the integration of an equivalent tree in set-of-possible-worlds representation with the same certain tree.

The data integration function integrate takes two parameters D_1 and D_2. It returns the integration result as a probabilistic XML tree. In the diagrams below, we have omitted probability and possibility nodes whenever there is only one possibility.

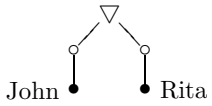
The example below shows how we can recursively integrate two certain trees.



After the first integration step, we obtain:

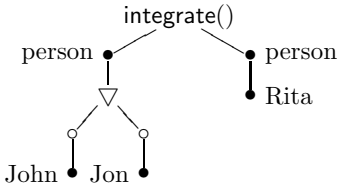


The second integration step integrate('John', 'Rita') obviously results in:

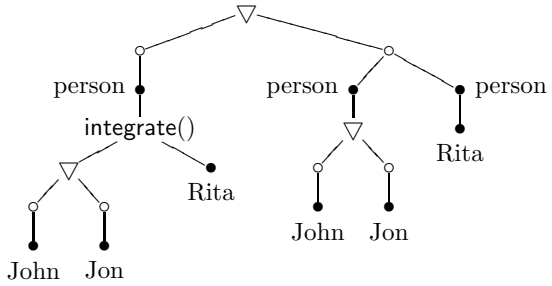


Observe the difference between integrating person elements, which are specified as being part of a sequence, and other elements including text nodes for which there can only be one. The former produces an additional possibility for the case that there exist two persons. In general, text nodes are also part of a sequence (e.g., paragraphs in a text document). Concatenating names of persons, however, does not make sense, so our rule engine decides that, for example, the name of a person can not be "JohnRita".

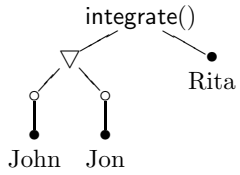
Integrating a probabilistic tree and a certain one proceeds as follows:



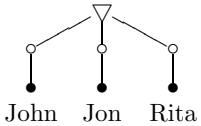
We would first integrate both person elements:



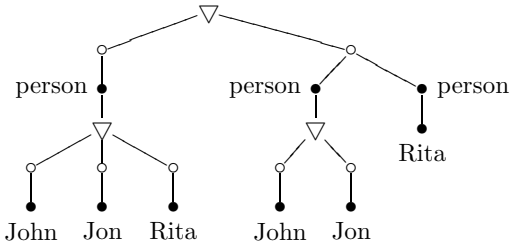
where



intuitively leads to



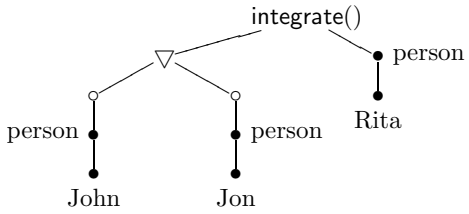
The entire resulting tree looks like:



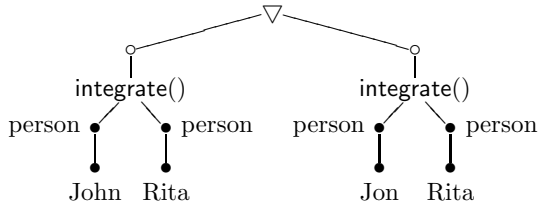
The resulting tree can be described as:

$$(\text{Rita} \vee \text{John} \vee \text{Jon}) \vee ((\text{John} \vee \text{Jon}) \wedge \text{Rita}) \quad (1)$$

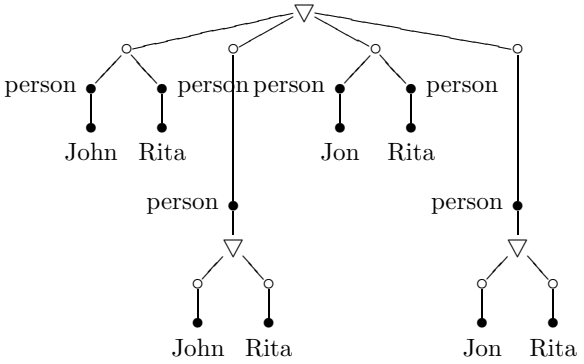
Moving the local possibility upwards in the tree, we get an equivalent less compact tree that is in all-possible-worlds representation. The integrate function now behaves as being applied to each possible world separately.



We integrate the person named "Rita" over both possibilities resulting in the following:



The final result is:



The boolean representation is

$$((\text{John} \wedge \text{Rita}) \vee (\text{John} \vee \text{Rita})) \vee ((\text{Jon} \wedge \text{Rita}) \vee (\text{Jon} \vee \text{Rita})) \quad (2)$$

Note that this is equivalent to the earlier obtained boolean representation. The trees are equivalent.

5.4 Assigning probabilities and confidence scores

As explained earlier, the rule engine determines in some way the probability of the various possibilities. Intelligence and the use of information from schema and other data sources can be used to limit the number of possibilities, but also to better assign probabilities.

The most simple scheme for assigning probabilities would be the following. Whenever the database conflicts with a foreign document one some element, we assign probability .5 to each resulting possibility. This approach has a severe drawback. For example, when a mobile device has once heard from another device that a person's name is "John" and it meets another device which says its name is "Jon", the scheme assigns the possibilities "John" and "Jon" each a probability of .5. But, when it has already heard from ninety-nine different devices, that the name of the person is "John", it should be very suspicious when it meets a device that says this person's name is "Jon". Therefore, it should give the possibility "Jon" a very small probability of, say, .01.

The rather basic rule engine of our prototype has a still simple but sufficient scheme for assigning probabilities. It is based on the premise that what you have seen twice, is twice as likely to be correct. In other words, a *confidence score* should be kept in the data. This factor is an indication of how certain we are about the data, which helps in assigning probabilities when integrating new data. Every time a different device claims a certain possibility is true, the confidence score is increased by one.

6 Experiments

The aim of our experiments is to gain insight into the explosion of possibilities in data and query answers, into the growth in size of the data set, and into the effectiveness of measures to cope with both problems. We use the two test documents shown in Figure 5. Schema information the rule engine uses: each person has only one firstname, lastname, phone and room number and there is only one persons element containing multiple person elements. The implementation of the prototype is in Java.

Experiment 1a. Integrating the test documents results in a probabilistic XML document with 3201 possible worlds. The least compact way of representing the document is in its all-possible-worlds representation (see Section 5.3). In this representation, the document is 2MB. A more compact representation (not the most compact way possible) results in a file size of 310kB. This is a compression factor of 6.6.

Experiment 1b. By dropping the rule that a person can only have one phone number, the number of possible worlds rises to 7105 and the file size (all-possible-worlds representation) grows to 4.4MB. The effect of the rule amounts to a factor of about 2.2 in number of both possible worlds and file size.

Document 1 (660 bytes)

```
<persons>
  <person>
    <firstname>Mark</firstname>
    <lastname>Hamburg</lastname>
    <phone>1010</phone>
    <room>3300</room>
  </person>
  <person>
    <firstname>Allen</firstname>
    <lastname>King</lastname>
    <phone>2020</phone>
    <room>3122</room>
  </person>
  <person>
    <firstname>Stan</firstname>
    <lastname>Choice</lastname>
    <phone>3030</phone>
    <room>3035</room>
  </person>
  <person>
    <firstname>John</firstname>
    <lastname>Friend</lastname>
    <phone>4040</phone>
    <room>3333</room>
  </person>
</persons>
```

Document 2 (366 bytes)

```
<persons>
  <person>
    <firstname>Mark</firstname>
    <lastname>Hamburg</lastname>
    <phone>1010</phone>
    <room>3301</room>
  </person>
  <person>
    <firstname>Allen</firstname>
    <lastname>Kingship</lastname>
    <phone>2020</phone>
    <room>3035</room>
  </person>
</persons>
```

Figure 5. Test documents

Experiment 2a: Query `//room[. = "3035"]`. In both test documents, room "3035" is occupied, but by a different person. Intuition tells us the query answer is certain, since there is a room "3035" in both test documents. In the integrated document, however, in only 3009 from the 3201 possible worlds (94%), someone actually occupies room "3035". This is due to the fact that our rule engine is not clever enough to, for example, exclude the possibility that Stan Choice is the same person as Mark Hamburg occupying room "3301" and John Friend is the same person as Allen Kingship occupying room "3333". In this possible world, nobody actually occupies room "3035". In 1041 of the 3201 possible worlds (32.5%), the room was occupied by two persons. Room "3301" is only occupied in the second test document. It was listed as occupied in 1633 of the 3201 possible worlds (51%).

Experiment 2b:

Query `//person[./firstname="John"]/room`. The person with name "John" appears in 2417 of the 3201 (75.5%) of the possible worlds. In 196 possible worlds, he occupies room "3301". In 196 possible worlds, he occupies room "3035". And in the remaining 2025 possible worlds, he can be found in room "3333".

Since we potentially get a possible query answer per possible world, there are 3201 possible answers:

- 2025 of the possible answers are "3333",
- 196 of the possible answers are "3035",
- 196 of the possible answers are "3301", and
- 784 of the possible answers are *empty*.

Note that simplifying this query answer will result in a probabilistic tree with only 4 possibilities. All uncertainty in the data not relevant for the query results in equal possible answers, that can be simplified easily.

7 Issues

This paper reports on a first step towards a probabilistic XML DBMS capable of data integration whereby it resolves conflicts by treating them as possibilities for how reality actually may be. We have made several simplifications, assumptions and other measures to be able to make this first step. The experience gave rise to a number of observations and problems. In this section, we would like to discuss some of these.

Evergrowing data set One of the prominent problems is the evergrowing data set. After every merge, new information and new possibilities are added, and nothing is ever forgotten. Especially in case of sequences being merged, the number of possibilities can be huge. For example, already for two small sequences of both 5 elements, there are 1546 distinct possibilities. It is, therefore, important to look at measures to keep the size of the data set within bounds. There are several possible measures that have to be investigated:

- Make the rule engine more intelligent by using schema and data information.
- Involve the user to rule out possibilities whenever there is an opportunity.
- Forget possibilities with small probability.

Using schema and data information We already mentioned that the rule engine can use schema information and other data sources to rule out certain possibilities. For example, cardinality or other schema constraints rule out those possibilities that violate those constraints. Integrity constraints in a similar way may rule out possibilities. More advanced techniques for data analysis can also be used. For example, data analysis on the contents of certain attributes or elements can establish that a certain content is highly unlikely, because it doesn't conform to the observed pattern. Machine learning techniques may be used similarly to [4]. For example, an ⟨age⟩-element will contain numbers between 0 and, say, 150. Statistical analysis on these elements may come to such conclusion, and hence, decide that a value of 1000 is highly unlikely.

Involve the user Data integration itself should not require the involvement of the user, but the user will look at the data in some future moment. For example, he/she looks up a person's address or telephone number. At that time, the user is capable of deciding what possibility does or does not correspond with reality. Furthermore, the user is already in interaction with the system, so that is a nice opportunity to involve him/her in ruling out possibilities.

In an ambient environment where sensors and other mechanisms monitor the user and his/her environment, information about which possibility actually holds in the real world, may be obtained without involvement of the user. For example, a user uses his/her mobile phone to call John. The Phone-application will first determine the phone number of John. According to Figure 4, the phone number is either 1111 or 2222 or there are two persons named John. If the phone actually gets connected and the phone call takes longer than one would expect from a wrong connection, the Phone-application may deduce that 1111 is a correct phone number for John, so the possibility of 2222 may be removed. Note that the possibility that there are two persons named John still exists. This way of ruling out possibilities obviously does not lie in the near future, but this is the kind of behaviour research on ambient intelligence is aiming for.

Forget possibilities with small probability If, after repeated integration with different data sources, a certain possibility does not get confirmed, its probability goes down. If it gets below a certain threshold, it becomes very likely

that it is simply an error. Furthermore, a possibility with a small probability will not have significant contribution to query results.

Since the probabilities of all possibilities belonging to one probability node should add up to 1, it is not straigforward to properly forget possibilities. Normalizing probabilities may have undesired consequences. There are, however, more advanced models of representing uncertainty. For example, the model of [1] contains the notion of '$*$', an unspecified rest probability representing the degree of *ignorance*. It allows one to assign one probability to all remaining possibilities without storing the details about these possibilities. One often only assumes a certain probability *distribution* to simplify querying in the presence of '$*$'. Note that in this way, one distances oneself from the closed world assumption. Perhaps, an approach based on an open world assumption is more appropriate on the whole.

Relational probabilistic databases If we compare our experiences with *relational* probabilistic databases, we observe that with XML it is much easier. First, the distinction between Type-1 and Type-2 disappears. Second, dependencies like mutual exclusiveness and simultaneous occurrence can be expressed naturally. Finally, XML's tree structure more naturally matches decision trees, which is basically what we are doing here.

Order Another issue is the issue of order in sequences. Given two sequences to be merged from two distinct data sources, the number of possibilities for merging these when order matters and is uncertain, grows even faster than what we have seen here, where we disregarded order.

Application interface In Section 4, we noted that it might be more logical to applications to get a query answer in the form of a set of possible answers. On the other hand, this is not an optimal query interface with respect to performance. What a good query API for applications looks like, is still an open issue. How an application can deal with the uncertainty it is confronted with, is another.

Confidence scores Section 5.4 presents a simple scheme for assigning probabilities during data integration. Confidence scores should be associated with elements, but, in this way, they obfuscate the original XML tree. A probabilistic XML document is assumed to represent a closed world. Within a closed world, confidence scores are not needed, because all possible worlds are *certain* worlds. But, when data integration takes place, the closed world assumption is dropped momentarily to allow for the foreign data to be integrated into the existing data. From a software design perspective, it may be better to store confidence scores separately. Furthermore, more advanced models for determining the confidence in certain parts of the data can obviously be developed.

Intelligent agent technology Finally, observe that the data management system behaves much like an intelligent agent. It tries to gather information on the real world on its own, but uses information it retrieves from other agents. We made the assumption that those other agents deliver information according to the proposed approach in an honest and predictable way. Without this assumption, other agents may have other intentions such as promoting an untruth, or sabotaging others. To be able to deal with such situations, a rather elaborate 'social' model is necessary where information on reliability and other properties of other agents is also maintained and shared among agents.

8 Conclusion

We presented an approach towards unattended XML data integration to be used in a mobile and ambient environment. The basic idea is that, without luxury of interaction with a user, a data manager should drop the assumption that it can always make correct choices when confronted with conflicting data. Therefore, we take the approach of allowing uncertainty in the data, which requires a data manager capable of handling such data properly.

The paper reports on a first step towards a probabilistic XML data management component. A formalization of important concepts like probabilistic tree and possible world is given. An approach for querying uncertain *relational* data that we proposed in earlier work [2], was adapted for XML context. A practical approach towards data integration based on probabilistic trees has been sketched. In this first step, we made several simplifying assumptions and abstractions. Nevertheless, we achieved our goal, an approach that allows for integrating data in an unattended manner. Throughout the text, we provided some information about a prototype that we have built to validate our ideas.

Obviously, there are several problems and possible improvements. We outlined several in Section 7 most notably the problem of the explosion of possibilities, hence, the problem of a fast and evergrowing data set. It is on this aspect, that we will focus next. Future work also includes investigating techniques for *efficient* data integration and querying, establishing the quality of query results, investigating how applications should interface with a probabilistic data manager, incorporating schema integration, investigating the benefit of other models for representing uncertainty, and investigating 'social models' to become more robust against malicious agents.

References

[1] D. Barbará, H. Garcia-Molina, and D. Porter. A probabilistic relational data model. In F. Bancilhon, C. Thanos, and D. Tsichritzis, editors, *Advances in Database Technology - EDBT'90. International Conference on Extending Database Technology, Venice, Italy, March 26-30, 1990, Proceedings*, volume 416 of *Lecture Notes in Computer Science*, pages 60–74. Springer, 1990.

[2] A. de Keijzer and M. van Keulen. A possible world approach to uncertain relational data. In *SIUFDB-04 International Workshop on Supporting Imprecision and Uncertainty in Flexible Databases, Zaragoza, Spain*, Sept. 2004.

[3] D. Dey and S. Sarkar. A probabilistic relational model and algebra. *ACM Trans. Database Syst.*, 21(3):339–369, 1996.

[4] A. Doan, P. Domingos, and A. Y. Halevy. Reconciling schemas of disparate data sources: A machine-learning approach. In *ACM SIGMOD Conference*, May 2001.

[5] T. Eiter, T. Lukasiewicz, and M. Walter. A data model and algebra for probabilistic complex values. *Annals of Mathematics and Artificial Intelligence*, 33(2-4):205–252, 2001.

[6] D. Florescu, D. Koller, and A. Levy. Using probabilistic information in data integration. In *Proceedings of the 23rd VLDB Conference*, 1997.

[7] N. Fuhr and T. Rölleke. A probabilistic relational algebra for the integration of information retrieval and database systems. *ACM Transactions on Information Systems*, 15(1):32–66, 1997.

[8] E. Hung. ProbSem: A probabilistic semistructured database model. Technical report, University of Maryland, 2002.

[9] E. Hung, L. Getoor, and V. Subrahmanian. PXML: A probabilistic semistructured data model and algebra. In *Proceedings of the 19th International Conference on Data Engineering (ICDE)*, 2003.

[10] L. V. S. Lakshmanan, N. Leone, R. Ross, and V. S. Subrahmanian. ProbView: a flexible probabilistic database system. *ACM Transactions on Database Systems*, 22(3):419–469, 1997.

[11] A. Nierman and H. V. Jagadish. ProTDB: Probabilistic data in XML. In *Proceedings of the 28th VLDB Conference*, 2002.

[12] E. Zimanyi and A. Pirotte. Imperfect information in relational databases. *Uncertainty Management in Information Systems, A. Motro and P. Smets, Eds.*, 1997.