# Querying and Updating Probabilistic Information in XML

Serge Abiteboul and Pierre Senellart

INRIA Futurs, 2-4 rue Jacques Monod, F-91893 Orsay Cedex, France LRI, Université Paris-Sud, F-91405 Orsay Cedex, France
serge.abiteboul@inria.fr, pierre@senellart.com

**Abstract.** We present in this paper a new model for representing probabilistic information in a semi-structured (XML) database, based on the use of probabilistic event variables. This work is motivated by the need of keeping track of both confidence and lineage of the information stored in a semi-structured warehouse. For instance, the modules of a (Hidden Web) content warehouse may derive information concerning the semantics of discovered Web services that is by nature not certain. Our model supports both querying (tree pattern queries with join) and updating (transactions containing an arbitrary set of insertions and deletions) over probabilistic tree data. We show that the model is as expressive as the impractical possible worlds model and that it is more expressive than a simple model where probability (confidence) is stored at each node of the tree. We briefly discuss implementation issues.

## 1 Introduction

The Web is developing very rapidly. However, the exploitation of the invaluable information resources it provides still mostly relies on humans overwhelmed by its dimension. If the problem of discovering information on the surface Web is facilitated by a number of directories and search engines, the discovery, understanding and use of databases published on the Web (typically via forms and Web services) is still particularly cumbersome. It is therefore most important to develop automatic tools to capture the semantics of this so-called *Hidden Web*. Such tools require a combination of techniques, for instance from information extraction, natural language processing, semantic analysis. We are concerned with combining such tools to develop a content warehouse to manage tree data coming from both the surface and Hidden Web [1]. Since such a system relies on techniques that are by nature imprecise, we need a model for imprecise tree data. Indeed, the main contribution of this paper is such a model, namely the *fuzzy tree model*.

Models for managing imprecise information are not new. In particular, a large literature exists for the relational model typically based on two approaches: (*i*) a probabilistic approach, e.g. [2, 3], and (*ii*) a logic approach, e.g. [4]. An originality of our model is that it is based on a tree model, primarily to meet the needs of the standard for data exchanges, XML [5]. Another originality is that our model

combines the two aforementioned approaches. First, probabilities are attached to pieces of data to capture the confidence the warehouse may have about the semantics of such pieces. Second, we rely on probabilistic events that are in the spirit of the logical conditions of [4], e.g. to capture choices performed during the extraction of information or its analysis. These probabilistic events capture dependencies between the probabilities of distinct pieces of data.

Although this model may seem complex, it turns out that it arises naturally when considering the management of information discovered on the Web. The model is at the same time expressive (it captures complex situations) and concise (it provides compact representations that can be processed). This will be shown by comparing the fuzzy tree model with two more standard models for describing imprecision. Most importantly, we will show that the model supports queries (tree pattern queries with joins, a standard subset of XQuery) and updates. Queries provide the means for a user to obtain information. Updates form the core component for the building of knowledge. The global system consists in a number of modules (crawlers, classifiers, data extractor, etc.). These tools introduce probabilistic knowledge in the content warehouse by updating it.

We briefly discuss an on-going implementation that supports the fuzzy tree model with both querying and updating.

The paper is organized as follows. In Section 2, we present more motivation. In Section 3, we briefly present preliminary notions used throughout the paper. Sections 4 and 5 discuss two simple models. The fuzzy tree model is the topic of Section 6. Before discussing related work and concluding, we present in Section 7 an on-going implementation of the fuzzy tree model.

## 2 Motivation

Since its creation in 1991, the World Wide Web has considerably grown, with billions of freely accessible pages. (See, e.g., [6].) It is generally assumed that the Hidden Web (also known as Deep Web or Invisible Web) that consists of databases accessible through HTML forms or Web services, is even much larger (see, e.g., [7]) and that its content is typically of better quality than that of the surface Web (e.g. *Yellow Pages*, U.S. *Census Bureau* ...). The Hidden Web is undoubtedly a potentially invaluable source of information.

We want to provide access to the Hidden Web via high-level queries. To illustrate, consider a Web user interested in available options for a particular car. A current search engine will return a list of HTML pages including the constructor's website. We would like our system to discover that a particular form on this site provides the desired information, fill that form, get the answer and directly provide the desired information. Such a semantic interpretation of the Hidden Web implies that the system has to discover the service, understand it, and index it so that when the query arrives, it can be correctly answered. The different main components of this process are illustrated in Figure 1. Our way of seeing this process is as follows. A number of modules (*independent agents*)

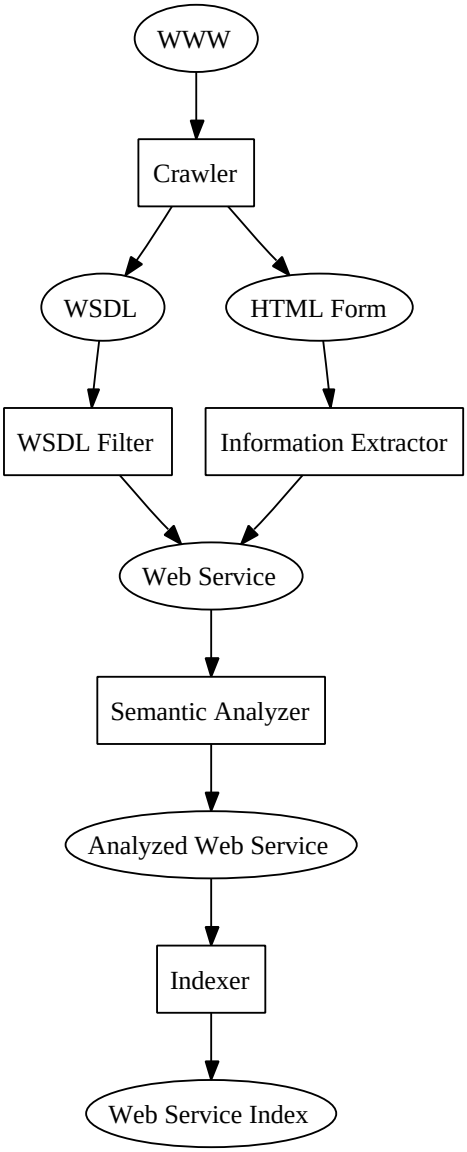



**Fig. 1.** Process for the Semantic Interpretation of the Hidden Web

process some XML data (the information we already have about the Hidden Web services) and keep enriching a content warehouse that holds that knowledge.

It is essential to observe that imprecision is unavoidable: the inference performed by most modules typically involves some level of confidence. For instance, the system may be rather confident (but not certain) that a site is that of the constructor. The system may have found more than one forms that may provide the options for a given make, but none that it is sure of. Any module participating in the construction of the warehouse must then be able to insert, modify or delete information *with a given confidence*. This confidence or, in other words, the *probability* that this information is true, should be handled throughout the entire process. It will be essential for informing the user of the confidence of portions of an answer as well as for ranking query results. Note that the process does not have to be sequential as shown in Figure 1 for simplicity. For instance, information extraction on one part of a HTML page may benefit of the semantic analysis of another part.

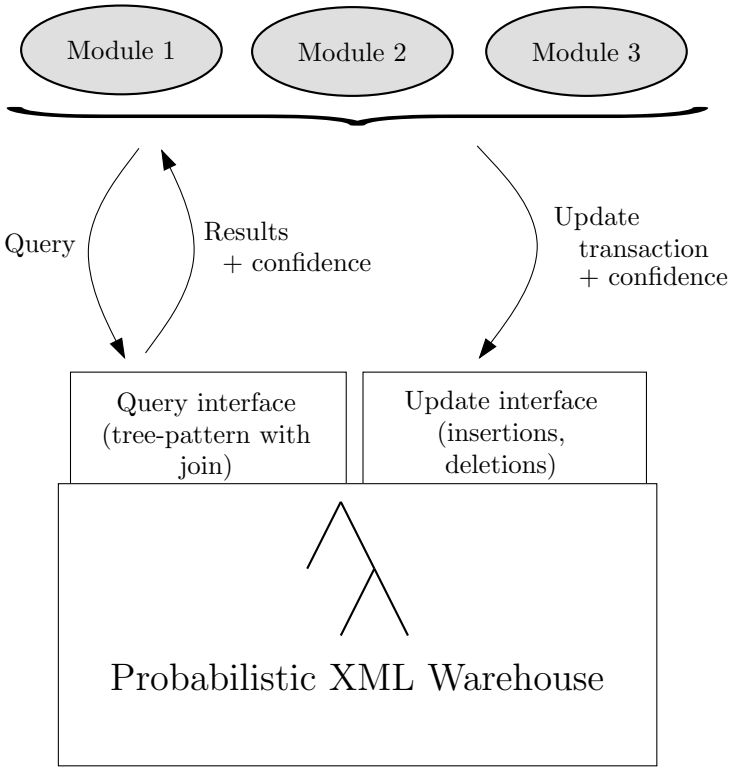



**Fig. 2.** Queries and Updates on a Probabilistic XML Warehouse

Three aspects, namely (*i*) the independent agents, (*ii*) the need to monitor the derivation of knowledge, and (*iii*) the non-sequentiality of the entire process, suggest the design of a *content-centric* system, as shown on Figure 2 in the style of [1]. More precisely, the system is built on top of a content (semi-structured) warehouse, with querying and updating capabilities supporting imprecise information. Probabilistic information is stored in the warehouse and confidence tracking is directly provided through the query and update interfaces. Each query result and update transaction comes with confidence information.

The purpose of this paper is to detail the model used to describe, query and update the probabilistic information stored in the warehouse. To conclude this section, we want to stress that the setting of Figure 2 is not restricted to a Hidden Web warehouse but is indeed much more general. Similar needs are encountered, for instance, when performing data integration [8], information extraction or automatic translation from natural language [9].

## 3 Preliminary definitions

In this section, we present preliminary definitions that are used in remaining of the paper. We assume the existence of a set $N$ of labels and a set $V$ of values (say, the set of strings, with a particular value $\varepsilon$, the empty string).

Although we are typically interested in XML, the standard for data exchange on the Web, we consider a simpler tree model that in particular ignores siblings ordering. We will mention in Section 7 how to capture more of XML.

**Definition 1.** *An (unordered)* data tree *$t$ is a 5-uple $t = (S, E, r, \varphi, \nu)$ where:*

- *$S$ is a finite set of nodes and $E \subseteq S^2$ a tree rooted in $r \in S$;*
- *$\varphi : S \to N$ associates a* label *to each node in $S$.*
- *$\nu$ associates a value in $V$ to leaves of the tree (where a* leaf *is a node in $S - \{r\}$ that has no child.)*

Examples of data trees are given in Figure 4. Nodes are marked with an oval containing the label. Observe that the node IDs (values in $S$) are not specified since they contain no semantics. Indeed, we will not distinguish between two trees that are the same up to renaming of these IDs.

The queries we use form a rather standard subset of XQuery [10], namely Tree-Pattern-With-Join. However, since there is no well accepted definition for such queries and, indeed, different definitions are used by different authors, we define them next formally.

**Definition 2.** *A* Tree-Pattern-With-Join (TPWJ) query *is a 3-uple $(t, \mathcal{D}, J)$ where:*

- *$t = (S, E, r, \varphi, \nu)$ is a data tree.*
- *$D \subseteq E$ is a set of* descendant edges *(the other edges are interpreted as* child edges*).*

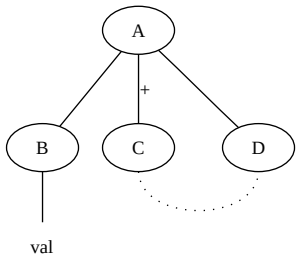



**Fig. 3.** Tree-Pattern-With-Join Query Example

– $J \subseteq S^2$ *is a set of* join conditions*, such that for all* $(s, s')$ *in* $J$*,* $s$ *and* $s'$ *are two leafs of* $t$ *and* $s \neq s'$*.*

Such a TPWJ query is represented as a graph as in Figure 3. Note the join condition between the nodes labelled $C$ and $D$. Intuitively, this query will be matched by a tree having, in particular, a child of its root labelled $D$ and a descendant of the root labelled $C$. The notion of *matching* of data tree by a TPWJ query is defined in a classical way:

**Definition 3.** *Let* $Q = (t, D, J)$ *with* $t = (S, E, r, \varphi, \nu)$ *be a TPWJ query and* $t' = (S', E', r', \varphi', \nu')$ *a data tree. Then a* valuation $\Psi$ *(from* $Q$ *in* $t'$*) is a mapping from* $S$ *to* $S'$ *verifying:*

*(i)* (root) $\Psi(r) = r'$.
*(ii)* (labels) $\forall s \in S, \varphi(\Psi(s)) = \varphi(s)$.
*(iii)* (edges)$\forall (s_1, s_2) \in E, if (s_1, s_2) \in D$*,* $\Psi(s_2)$ *is a descendant of* $\Psi(s_1)$*, otherwise it is a child of* $\Psi(s_1)$.
*(iv)* (values) *For each leaf* $s$ *of* $t$ *with* $\nu(s) \neq \varepsilon$*,* $\Psi(s)$ *is a leaf of* $t'$ *and* $\nu'(\Psi(s)) = \nu(s)$.
*(v)* (join conditions) *For each* $(s1, s2) \in J$*, both* $\Psi(s1)$ *and* $\Psi(s2)$ *are leaves of* $t'$ *and* $\nu'(\Psi(s_1)) = \nu'(\Psi(s_2))$.

*Let* $\Psi$ *be such a valuation. Then the minimal subtree of* $t'$ *containing* $\Psi(S)$ *is called an* answer *(we consider that a subtree is defined by a connected subset of the set of nodes containing the root). The set of all answers is denoted* $Q(t')$.

Figure 4 gives an example of a valuation from the query of Figure 3 to a simple data tree. Observe that neither the second $B$ node nor the $F$ node is part of the answer, by definition of minimal subtree.

We next define here update operations, whose basic components are *insertions* and *deletions*. Let $t = (S, E, r, \varphi, \nu)$ and $t' = (S', E', r', \varphi', \nu')$ be two data trees. Assume without loss of generality that they use different IDs, i.e. $S \cap S' = \emptyset$. An *insertion* is an expression $i(t, n, t')$ where $n$ is in $S$, the node where $t'$ is to be inserted. The result of the insertion is the data tree: $v = (S \cup S', E \cup E' \cup \{n, r'\}, r, \varphi \sqcup \varphi', \nu \sqcup \nu')$.

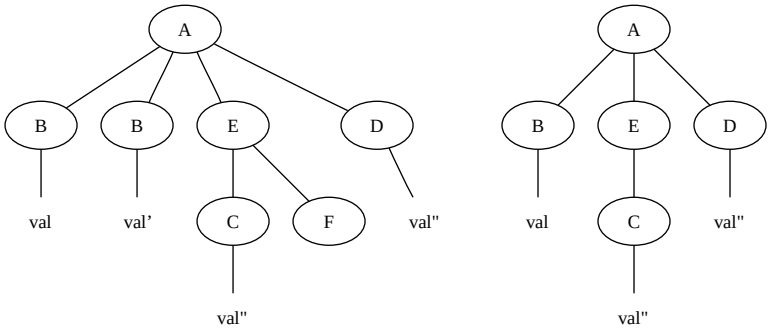



**Fig. 4.** Matching of the query of Figure 3 on an example tree

A *deletion* is an expression $d(t, n)$ where $n$ is in $S - \{r\}$. Node $n$ is removed as well as all its descendants. The formal definition is straightforward and thus omitted.

Insertions and deletions are *elementary* updates that are used to define *update transactions*. Typically, one want to perform a number of update operations based on the result of a query. This motivates the following definition.

**Definition 4.** *An* update transaction *is a pair* $\tau = (Q, U)$ *where:*

- $Q = (t_Q, D, J)$ *is a TPWJ query.*
- $U$ *is a set* $(i_1 \dots i_p, d_1 \dots d_q)$ *where* $i_1 \dots i_p$ *are insertions on* $t_Q$ *and* $d_1 \dots d_q$ *are deletions on* $t_Q$*.*

Queries are used to select the nodes of the trees where insertions or deletions are made. Intuitively, when one applies a transaction on a data tree $t$, one operation, say $d_i$, results in the deletion of a subtree for each valuation of $Q$. We assume that the various trees that are inserted all use fresh IDs.

**Definition 5.** *Let* $\tau = (Q, U)$ *be an update transaction.*

*Let* $t$ *be a data tree matched by* $Q$ *and let* $\Psi_1 \dots \Psi_n$ *be the valuations of* $Q$ *on* $t$*.*

*Let* $(n^{i_1} \dots n^{i_p}, n^{d_1} \dots n^{d_q})$ *be the nodes of* $Q$ *of the insertions and deletions of* $U$*.*

*For each* $k$ *with* $1 \leq k \leq p$*, we define the set* $I_k = \bigcup_{1 \leq j \leq n} \{\Psi_j(n^{i_k})\}$*. For each* $k$ *with* $1 \leq k \leq q$*, we define the set* $D_k = \bigcup_{1 \leq j \leq n} \{\Psi_j(n^{d_k})\}$*.*

*The result of the transaction* $\tau$ *on* $t$*, denoted* $\tau(t)$*, is the result of the insertions* $i_1 \dots i_p$ *on, respectively, each of the nodes of* $I_1 \dots I_p$ *and the deletions* $d_1 \dots d_q$ *on, respectively, each of the nodes of* $D_1 \dots D_q$*.*

We are now ready to consider models for probabilistic tree data.

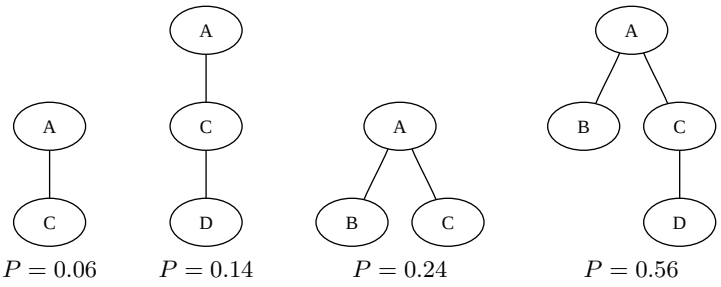



**Fig. 5.** Example Possible Worlds set

## 4 Possible Worlds Model

A natural way of representing probabilistic information is to list all *possible worlds*, each with its probability. See the example in Figure 5. More formally:

**Definition 6.** *A Possible Worlds (PW) set $T$ is a finite set of pairs $(t_i, p_i)$ where each $t_i$ is a data tree, each $p_i$ is a positive real and $\sum_{i=1}^{n} p_i = 1$.*

If $(t, p)$ is in a PW set $T$, this means that there is a probability $p$ that the information contained in $T$ is indeed $t$. This is the rather general way of representing probabilistic semi-structured information. In complex cases, one may have a very large number of possible worlds and clearly, it would be more practical to have a more compact representation of sets of possible worlds; indeed, we will propose some in the next sections. But for now, let us consider the Possible World sets and a particular issue, namely normalization.

A PW set $T = \{(t_i, p_i)\}$ is said to be *normalized* if there is no $i, j$ distinct such that $t_i, t_j$ are identical (up to node isomorphism). The *normalization* of a PW set $T$ is obtained by regrouping the identical component trees into one component and summing their probabilities. For instance, consider $T = \{(t_1, 1/3), (t_2, 1/3), (t_3, 1/3)\}$ where $t_1, t_3$ are identical up to node renaming. Then the normalization of $T$ is the PW set $\{(t_1, 2/3), (t_2, 1/3)\}$. In the remaining of this paper, we will assume that all PW trees are normalized.

We extend this notion of normalization to sets of $(t_i, p_i)$ pairs where the $p_i$ do not sum to 1.

The definition of TPWJ query on a data tree can be extended in a quite natural way to PW sets.

**Definition 7.** *Let $Q$ be a TPWJ query and $T = \{(t_i, p_i)\}$ a PW set. The set of* probabilistic possible answers *is:*

$$\mathcal{P} = \{(t, p_i) \mid (t_i, p_i) \in T, t \in Q(t_i)\}$$

*The* result *of $Q$ for $T$ is the normalization $Q(T)$ of $\mathcal{P}$.*

Note that by construction $Q(T)$ is not always a PW set since the sum of probabilities is not 1 in the general. The fact that $(t, p) \in Q(T)$ is interpreted as there is a probability $p$ that $t$ is a result of the query $Q$ over $T$. So, for instance, the query "Who are the children of John?" may return "Alice" with a probability 0.9 and "Bob" with a probability 0.4. Note that the sum is greater than 1.

Now consider updates. By having defined an update transaction with a TPWJ query matching the tree and not by an absolute position in the tree, it is possible to extend this definition to PW sets.

A *probabilistic update transaction* is a pair $(\tau, c)$ where $\tau$ is an update transaction and $c \in ]0; 1]$ is the *confidence* we have in the transaction. Now we have:

**Definition 8.** *Let $T = \{(t_i, p_i)\}$ be a PW set, $(\tau, c)$ a probabilistic update transaction, $\tau = (Q, Seq)$, the result of $(\tau, c)$ on $T$, denoted $(\tau, c)(T)$, is a PW set obtained by normalizing:*

$$\begin{array}{l} \{(t,p) \in T \mid t \text{ is not selected by } Q\} \\ \bigcup \{(\tau(t), p \cdot c) \mid t \text{ is selected by } Q\} \\ \bigcup \{(t, p \cdot (1-c)) \mid t \text{ is selected by } Q\} \end{array}$$

In other words, the possible worlds set resulting from an update transaction with confidence $c$ of a component $t$ of a possible worlds set $T$ contains both the original tree $t$, with probability multiplied by $(1-c)$, and the tree resulting from the transaction, with probability multiplied by $c$. This implicitly supposes that all update transactions are independent.

Note that in the worst case, the number of components is multiplied by 2. This may occur for instance if one inserts a node as a child of the root. Then $Q$ matches all data trees. Thus, the number of components grows, in the worst case, exponentially in the number of update transactions performed on the PW set.

To conclude this section, note that the PW model is not practical storage-wise. It is neither practical for query and update processing (in particular because of the potential exponential explosion). We next look at alternative ways of representing probabilistic tree information. The possible world semantics is natural and will provide guidelines for these more complex models.

## 5 Simple Probabilistic Tree Model

In the spirit of probabilistic models for the relational model, we can attach a probability to each node in the tree. This is the basis of the model studied in [11]. The intuition is that it captures the probability of that node to be present assuming its parent is. A limitation of this model is that the only probability dependency that is captured is between the dependencies of nodes in a parent/child relationship. We study next this model and highlight limitations that will motivate the fuzzy tree model of the following section.

**Definition 9.** *A* Simple Probabilistic (SP) tree *$T$ is a pair $(t, \pi)$ where $t = (S, E, r, \varphi, \nu)$ is a data tree and $\pi : S - \{r\} \rightarrow ]0; 1]$ assigns probabilities to tree nodes.*

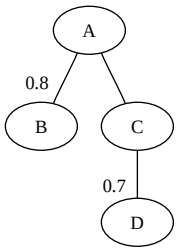



**Fig. 6.** Example Simple Probabilistic tree

Such an SP tree is represented as in Figure 6. Only probabilities non set to 1 are shown.

We can give a *possible worlds semantics* to an SP tree as follows. Choose an arbitrary $X \subseteq S$. Consider $t_X$ the tree obtained by removing from $t$ all nodes not in $X$ (and their descendants). We assign to this tree, the probability:

$$p_X = \prod_{s \in X} \pi(s) \times \prod_{s \in S-X} (1 - \pi(s))$$

The possible world semantics of $T$, denoted $Sem(T)$, is defined as:

$$normalization(\{(t_X, p_X) \mid X \subseteq S\})$$

Note that, as $X$ is an arbitrary subset of $S$, a given tree $t'$ can be obtained from various subsets $X$, including subsets which contain nodes not in $t'$. The normalization, by summing on the different $X$ leading to the same tree, ensures that the probability of $t'$ is correct. In particular, this definition correctly provides a PW set because:

$$\sum_{X \subseteq S} (\prod_{s \in X} \pi(s) \times \prod_{s \in S-X} (1 - \pi(s))) = 1$$

from classical probability equations.

As an example, the semantics of the SP tree of Figure 6 is the PW set on Figure 5. A natural question is then whether the SP tree model is as expressive as the PW tree model. The answer is no.

**Proposition 1.** *There exists a PW set which is not the PW semantics of an SP tree.*

Figure 7 is an example of a PW set that has no equivalent SP tree. Intuitively, an equivalent SP tree would necessarily have nodes $A$, $B$, $C$ and $D$, so the PW set would contain a tree with these 4 nodes, which it does not have, a contradiction.

This negative result is not a sufficient reason for rejecting the SP model, since one might argue that PW trees not representable in the SP model are of little practical interest. Let us consider queries and updates in the SP model.

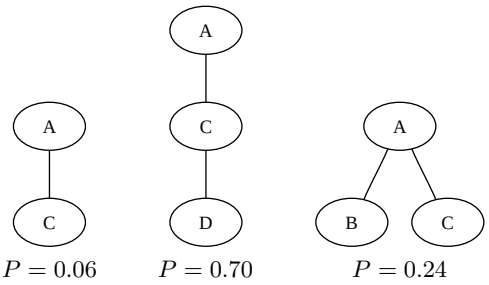



**Fig. 7.** More complex Possible Worlds tree

**Definition 10.** *Let $Q$ be a TPWJ query and $T = (t, p)$ an SP tree. The result of $Q$ on $T$, denoted $Q(T)$, is the normalization of the set:*

$$\bigcup_{u \in Q(t)} \{(u, \prod_{n \text{ node of } u} p(n))\}$$

Once again, $Q(T)$ is *not* in general a PW set. The way to read $(t', p') \in Q(T)$ is that there is a probability $p'$ that $t'$ is a result to $Q(T)$. We have the following result, which shows that the definition is coherent with the PW semantics of an SP tree:

**Theorem 1.** *Let $Q$ be a TPWJ query. Then $Q(Sem(T)) = Q(T)$.*

Now consider updates. The following result demonstrates that the SP tree model does not meet our requirements in terms of update, so motivates the last model we consider in the next section.

**Proposition 2.** *There exists an SP tree $T$ and a probabilistic update transaction $(\tau, c)$ such that there is no SP tree whose semantics is $(\tau, c)(Sem(T))$.*

In other words, "SP trees are not closed under transactions". This comes from the fact that dependencies between nodes are not expressible in the SP model. For instance, if an update transaction adds two nodes under one common node, their presence is completely correlated whereas the presence of siblings is independent in the SP model. Indeed, a simple *modification* that can be seen as an interdependent succession of an insertion and a deletion, cannot be represented in the SP model. Actually, the problem is even deeper: as the positions where the updates are performed are selected by a query, the update to be performed may be conditioned by the assumptions that were made to realize the query. There is no way to specify this kind of dependency.

We next present an other model that overcomes this limitation.

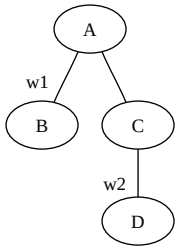
## 6 Fuzzy Tree Model

In this section, we propose an original model for representing probabilistic information in semi-structured databases, that we call the *fuzzy tree model*. This

model is inspired by the SP model from the previous section and enriches it using conditions à la [4], that are called probabilistic conditions here. We first introduce the model; then consider the possible world semantics; and finally queries and updates.

### 6.1 Model

The conditions we use are defined using the auxiliary concept of *probabilistic event*. Given a set $W$ of event names, a probability distribution $\pi$ assigns probabilities, i.e. values in $]0;1]$ to elements of $W$. An *event condition* (over $W$) is a finite (possibly empty) set of *event atoms* of the form $w$ or $\neg w$ where $w$ is an event in $W$.




| Event | Proba. |
|---|---|
| w1 | 0.8 |
| w2 | 0.7 |

**Fig. 8.** Example fuzzy tree

The intuition is that we assign probabilistic conditions to nodes in the trees, instead of assigning them simple probabilities like in the SP model. This mechanism captures complex dependencies between nodes in the database.

**Definition 11.** *A fuzzy tree $T$ is a 3-uple $(t, \pi, \gamma)$ where:*

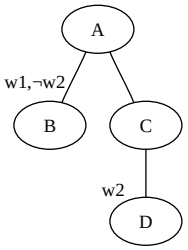
- *$t = (S, E, r, \varphi, \nu)$ is a data tree*
- *$\pi$ is some probability distribution over some set $W$ of events*
- *$\gamma$ assigns event conditions to nodes in $S - \{r\}$.*

Figures 8 and 9 show examples of fuzzy trees. True conditions (i.e. empty set of events) are not shown.

### 6.2 Possible Worlds Semantics

The semantics of a fuzzy tree is defined using PW sets as follows:

**Definition 12.** *Let $T = (t, \pi, \gamma)$ with $t = (S, E, r, \varphi, \nu)$ be a fuzzy tree. Let $W$ be the event names occurring in $T$. The possible worlds semantics of $T$ is the PW set, denoted $Sem(T)$, defined as the normalization of:*




| Event | Proba. |
|---|---|
| w1 | 0.8 |
| w2 | 0.7 |

**Fig. 9.** More complex fuzzy tree

$$\bigcup_{V \subseteq W} \{(t_{|V}, \prod_{w \in V} \pi(w) \prod_{w \in W-V} (1 - \pi(w)))\}$$

*where $t_{|V}$ is the subtree of $t$ where all nodes conditioned by a '$\neg w$' atom with $w \in V$ or a '$w$' atom with $w \notin V$ are removed (as well as their descendants).*

Example of PW semantics of fuzzy trees are given in Figures 5 and 7, respectively for the fuzzy trees of Figures 8 and 9. Observe that the fuzzy tree model is more expressive than the SP model, since the latter did not have an equivalent of the PW set represented in Figure 7. Actually, the following important result states that the fuzzy tree model is as expressive as the PW model.

**Theorem 2.** *For each PW set $X$, there exists a fuzzy tree $T$ such that $X = Sem(T)$.*

To see the previous result, consider a PW set $X$. For each $(t_i, p_i)$ in $X$, consider an event $w_i$. Besides the root that is not conditioned, each node in the fuzzy tree has a condition of the form $\neg w_1, \neg w_2 \ldots \neg w_{i-1}, w_i, \neg w_{i+1} \ldots \neg w_n$. Obviously, this construction is very verbose since the resulting fuzzy tree has approximatively the size as the original PW set. It will be of course interesting to consider more compact equivalent fuzzy trees.

### 6.3 Querying

**Definition 13.** *The result of $Q$ on a fuzzy tree $T = (t, \pi, \gamma)$, denoted $Q(T)$, is the normalization of the set:*

$$\bigcup_{u \in Q(t)} \{(u, eval(\bigcup_{n \text{ node of } u} \gamma(n)))\}$$

*where $eval(cond)$ returns 0 if there is an event $w$ such that both '$w$' and '$\neg w$' are in $cond$, and $\prod_{w \in cond} \pi(w) \cdot \prod_{\neg w \in cond} (1 - \pi(w))$ otherwise.*

*When normalizing the set, if one of the probability is 0, the element is removed. If the resulting set is empty, $Q$ does not match $T$.*

Note that we define above the result of a query directly as a result in the possible world semantics. Although not done here, one could define it as a fuzzy tree whose semantics captures all the answers. The correctness of this definition is demonstrated in the following result:

**Theorem 3.** *Let $T$ be a fuzzy tree and $Q$ be a TPWJ query. $Q$ matches $T$ if and only if $Q$ matches $Sem(T)$. Moreover, $Q(T) = Q(Sem(T))$.*

### 6.4 Updating

Finally, we show that unlike the SP model, the fuzzy tree model supports arbitrary probabilistic update transactions. We present informally the definition of the way updates are carried out for fuzzy trees. The formal definition somewhat more involved is omitted.

Let $(\tau, c)$ with $\tau = (Q, U)$ be a probabilistic update transaction and $T = (t, \pi, \gamma)$ a fuzzy tree. Let $w$ be a fresh event variable.

If $Q(T) = \emptyset$, the result of $(\tau, c)$ on $T$ is obviously $T$. Consider now the case where $|Q(T)| = 1$, that is where the position of update operations is uniquely defined. Let $u$ be the unique element of $Q(T)$ and $cond = \bigcup_{n \text{ node of } u} \gamma(n)$. $cond$ is the set of conditions that should be applied to the inserted and deleted nodes. The result of $(\tau, c)$ on $T$, denoted $(\tau, c)(T)$, is the fuzzy tree where:

- Insertions are performed at the position mapped by $Q$ on $t$ in $u$. If $n$ is the position to insert a subtree $t'$, and $cond_{ancestors}$ is the union of the event conditions on the (strict) ancestors of $n$, $t'$ is inserted and its root is assigned the condition $\{w\} \cup (cond - (\gamma(n) \cup cond_{ancestors}))$.
- Deletions are performed at the position mapped by $Q$ on $t$ in the following way. Let $n$ be the node to be deleted and $cond_{ancestors}$ be the union of the event conditions on the (strict) ancestors of $n$. Let $cond_{new} = \{w\} \cup cond - (\gamma(n) \cup cond_{ancestors})$. Intuitively, the deletion of the node should be conditioned by $cond_{new}$; in other words, the node should remain if the disjunction of the negated atoms of $cond_{new}$ is true. More precisely, the original $n$ node is replaced by as many copies as elements of $cond_{new}$. Let $a_1 \dots a_p$ be the $p$ elements of $cond_{new}$. The first copy of $n$ is annotated with condition $\gamma(n) \cup \{\neg a_1\}$. The second copy of $n$ is annotated with condition $\gamma(n) \cup \{a_1, \neg a_2\}$. The third copy of $n$ is annotated with condition $\gamma(n) \cup \{a_1, a_2, \neg a_3\} \dots$ The last copy of $n$ is annotated with conditions $\gamma(n) \cup \{a_1 \dots a_{n-1}, \neg a_n\}$.

This handles the case where $|Q(t)| = 1$. Consider now what happens when a query returns more than one result:

- Insertions are performed at the positions mapped by any of the valuations of $Q$. This means that a subtree may be inserted twice under the same node, but with different conditions.

– Deletions are performed at the positions mapped by any of the valuations of $Q$. If two deletions occur at the same position, the deletion is conditioned by the *disjunction* of the two condition sets. Let $F$ be the corresponding logical formula. The node should remain if $\neg F$ is true. Let $G$ be the disjunctive normal form of $\neg F$. The node $n$ is replaced by as many nodes as terms in the disjunction $G$, each of them corresponding to one of this conjunctive term, as in the case where $|Q(t)| = 1$.

This definition, in particular the deletion part, is quite complex. Let us consider the example of the simple *deduplication* task on Figure 10. Here, the two $B$ nodes are considered as representing the same information with confidence 0.9; therefore, they are removed from the tree and replaced with a new unified node with the same probability. However, the first $B$ will not be deleted if the deduplication hypothesis is not true (w1,¬w3) or if the deduplication hypothesis is true but the second $B$ does not exist (w1,w3,¬w2). The same holds for the second $B$. This simple example shows that the information contained in the tree can become quite complex, all the more in the presence of deletions.

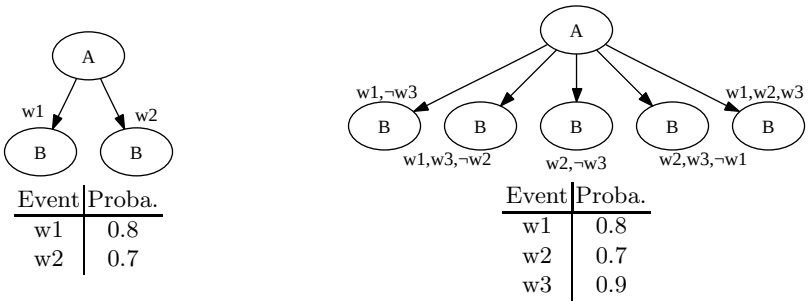



**Fig. 10.** Deduplication example

We now have:

**Theorem 4.** *Let $(\tau, c)$ be a probabilistic update transaction and $T$ a fuzzy tree. Then $Sem((\tau, c)(T)) = (\tau, c)(Sem(T))$.*

The fuzzy tree model provides a more concise representation of imprecision than the possible world model. Updates can be captured in this model. Complex operations may still be costly, but simple operations (insertions, or deletions without dependency on another branch of the tree) do not yield an exponential growth, as it is the case for the PW model. As for the conditional tables of [4], the "simplification" of a fuzzy tree, i.e. finding a fuzzy tree that is minimal in some sense, is rather complex. Its exact complexity is still open.

*Remark 1.* An interesting side benefit of using the fuzzy tree model is the possibility to keep *lineage* (or *history*) information about the data. Since every node is conditioned by event variables corresponding to update transactions, we can associate meta-data to these variables to record information about the origin (when? who? why?) of the corresponding transaction. Note that these variables

are preserved throughout the whole process and a fuzzy tree system would be able to deliver, along with query results and probabilities, information about the lineage associated with a piece of data (possibly updated more than once) and query results. This is an important component toward understanding why a given piece of data occurs in the data or in a query result.

## 7 Implementation

This section briefly discusses our implementation of a fuzzy tree system. The on-going implementation is based on the following components, and will soon be available at http://pierre.senellart.com/software/fuzzyxml/.

- XML documents are stored in a file system. (The use of an XML repository will be considered in the future.)
- TPWJ queries are represented as XML fragments.
- The query evaluation over fuzzy trees is implemented using the Qizx/open Java XQuery engine [12]. TPWJ queries are compiled into XQuery queries, whose results are converted to the minimal subtrees referred to in Definition 2. A post-processing is performed on the resulting subtrees to compute the associated probabilities.
- Query compilation uses a dataguide of the document obtained by using the XML Summary Drawer described in [13] for optimization.
- Updates are performed directly on the XML tree (cf Section 6.4).
- The complete system is made available as a webservice, to be integrated in more complex systems such as the one represented in Figure 2.

Observe that we need to record probabilistic meta-information in the XML documents. This is achieved by adding an element containing event conditions on every conditioned node, and maintaining external tables with the association between event names and probabilities. To conclude this section, we briefly discuss the limitations of our work when exposed to real-life XML documents:

- The trees we consider are unordered, which is absolutely inherent to the approach; we have to assume that the applications we will support are only concerned with queries that do not rely on the order of siblings. This is a rather standard assumption.
- Mixed content (nodes including both subnodes and textual content) is not yet supported. This can be easily fixed by adding virtual text nodes above every text fragment.
- Attribute nodes are not yet supported either. They can easily be replaced by standard nodes.

## 8 Related Work

The topic of probabilistic databases has been much studied, see for instance [4, 3, 14–16], and [2, 17] for more recent works. In [17], Widom stresses the need for

a system maintaining both *probability* and *lineage* of the data. In that paper, imprecision comes from three distinct sources: inaccuracy of the values, confidence in the tuples of the relations and incompletude in the coverage of relations. We were only interested here in this second source of imprecision. The idea of associating probabilistic formulas to node comes from the *conditional tables* of [4].

The semi-structured data model [18] and its concrete representation XML [5] are now widely used for exchanging data on the Web, where flexible schemas are preferred to the strict typing of the relational model. A relatively small number of works, however, have dealt with the representation of probabilistic semi-structured data. In [19], Dekhtyar and all use a semi-structured database to store complex probabilistic distributions of data which is essentially relational. Works closer to ours are [11, 20, 8]. Nierman et al [11] describe a variant of the SP model and present strategies for efficient evaluations of logical queries. In [20], a very complex model, based on directed acyclic graphs, is developed, along with an algebraic query language. Finally, Keulen et al [8] present an approach to data integration using probabilistic trees; their model is a mix of the PW and SP model, which allows both extensive descriptions of the possible worlds and node-based factorization. Querying and the way to present data integration results on this model are also shown. It is to be noted that none of the previous works, to the best of our knowledge, describes in an extensive way how to do updates on a probabilistic semi-structured database, which is one of the main contributions of this paper.

## 9 Conclusion

The work presented here is part of a larger project on the construction of content warehouses from (Hidden) Web resources as described in Section 2. After working for a while on the topic, we realized that imprecision had to be a core part of the XML-based warehouse since ad-hoc processing of imprecision simply did not scale. This observation motivated the present work. We need now to complete the implementation of the fuzzy tree system, move it to an efficient XML repository and experiment it in real conditions.

A most important direction of research is to develop optimization techniques tailored to the fuzzy tree model. In particular, one would like to cut off tree branches that provide low confidence data. Also, we want to study fuzzy tree simplification, i.e. finding more compact representations of imprecise data, possibly at the cost of some moderate loss of precision.

As it is defined, the fuzzy tree model is not completely algebraic: if the result of an update is a tree, the result of a query is a set of tree/probability pairs. It turns out that using a similar construction, one can also obtain a representation of the answer in terms of fuzzy trees. The details have to be worked out.

Finally, an interesting aspect is schema validation. Suppose we have a fuzzy tree representation $T$ of some data and also know that it is conform to some DTD $D$. Its semantics can be seen as $X = \{Sem(T) \cap sat(D)\}$ where $sat(D)$ is the set of documents validating $D$. An issue is to efficiently compute a fuzzy tree

for $X$. Of course, we will have to ignore order-related typing issues. But other aspects such as cardinalities are already quite challenging.

## References


1. Abiteboul, S., Nguyen, B., Ruberg, G.: Building an active content warehouse. In: Processing and Managing Complex Data for Decision Support. Idea Group Publishing (2005)
2. Dalvi, N.N., Suciu, D.: Efficient query evaluation on probabilistic databases. In: Very Large Data Bases, Hong Kong, China (2004) 864–875
3. de Rougemont, M.: The reliability of queries. In: Principles Of Database Systems, San Jose, United States (1995) 286–291
4. Imieliński, T., Lipski, W.: Incomplete information in relational databases. J. ACM **31** (1984) 761–791
5. World Wide Web Consortium: eXtensible Markup Language recommendation (2004) http://www.w3.org/TR/REC-xml/.
6. Google: Google search engine (2005) http://www.google.com/.
7. BrightPlanet: The Deep Web: Surfacing hidden value. White Paper (2000)
8. van Keulen, M., de Keijzer, A., Alink, W.: A probabilistic XML approach to data integration. In: International Conference on Data Engineering. (2005) 459–470
9. Attnäs, M., Senellart, P., Senellart, J.: Integration of SYSTRAN MT systems in an open workflow. In: MT Summit X, Phuket, Thailand (2005)
10. World Wide Web Consortium: XQuery 1.0: An XML Query Language (2005) http://www.w3.org/TR/xquery/.
11. Nierman, A., Jagadish, H.V.: ProTDB: Probabilistic data in XML. In: Very Large Data Bases, Hong Kong, China (2002)
12. Franc, X.: Qizx/open (2005) http://www.xfra.net/qizxopen/.
13. Arion, A., Bonifati, A., Manolescu, I., Pugliese, A.: Path summaries and path partitioning in modern XML databases. Technical Report 437, Gemo (2005)
14. Cavallo, R., Pittarelli, M.: The theory of probabilistic databases. In: Very Large Data Bases. (1987) 71–81
15. Barbará, D., Garcia-Molina, H., Porter, D.: The management of probabilistic data. IEEE Transactions on Knowledge and Data Engineering **4** (1992) 487–502
16. Fuhr, N., Rölleke, T.: A probabilistic relational algebra for the integration of information retrieval and database systems. ACM Trans. Inf. Syst. **15** (1997) 32–66
17. Widom, J.: Trio: A system for integrated management of data, accuracy, and lineage. In: Biennal Conference on Innovative Data Systems Research, Pacific Grove, USA (2005)
18. Abiteboul, S., Buneman, P., Suciu, D.: Data on the Web: from relations to semistructured data and XML. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2000)
19. Dekhtyar, A., Goldsmith, J., Hawkes, S.R.: Semistructured probabilistic databases. In: Statistical and Scientific Database Management, Tokyo, Japan (2001) 36–45
20. Hung, E., Getoor, L., Subrahmanian, V.S.: PXML: A probabilistic semistructured data model and algebra. In: International Conference on Data Engineering, Bangalore, India (2003) 467–478