Querying and Updating Probabilistic Information in XML

Serge Abiteboul and Pierre Senellart*

INRIA Futurs & LRI, Université Paris-Sud, 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, namely the fuzzy tree 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 highlight its expressive power and discuss implementation issues.

1 Introduction

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 HTML forms) is still cumbersome. It is therefore important to develop automatic tools to capture the semantics of this so-called *Hidden Web*. Such tools require a combination of techniques, e.g. from information extraction or natural language processing. 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. The main contribution of this paper is such a model, 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. Another originality is that our model combines the two aforementioned approaches. Probabilities are attached to pieces of data to capture the confidence the warehouse may have about the semantics of such pieces, while 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. Finally, while

* This work was developed in the framework of the RNTL Project WebContent.

Y. Ioannidis et al. (Eds.): EDBT 2006, LNCS 3896, pp. 1059–1068, 2006.

most works on incomplete information focus on queries, updates play a central role in the present work as, for instance, in [5].

The fuzzy tree model arises quite 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). 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.

The paper is organized as follows. In Sect. 2, we present more motivation. In Sect. 3, we briefly present preliminary notions used throughout the paper. Section 4 discusses two simple models. The fuzzy tree model is the topic of Sect. 5. Before discussing related work and concluding, we present in Sect. 6 an on-going implementation of the fuzzy tree model. **An extended version of this paper is available in [6].**

2 Motivation

Since its creation in 1991, the World Wide Web has considerably grown, with billions of freely accessible pages. 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. The Hidden Web is undoubtedly an 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 a given 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, at query time, the proper data may be retrieved.

Imprecision is inherent in such a process: 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. 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 the information is true, should be handled throughout the entire process. It is essential for informing the user of the confidence of portions of an answer as well as for ranking query results.

Three aspects, namely (*i*) the independent agents, (*ii*) the need to monitor the derivation of knowledge, and (*iii*) the non-sequentiality of the entire

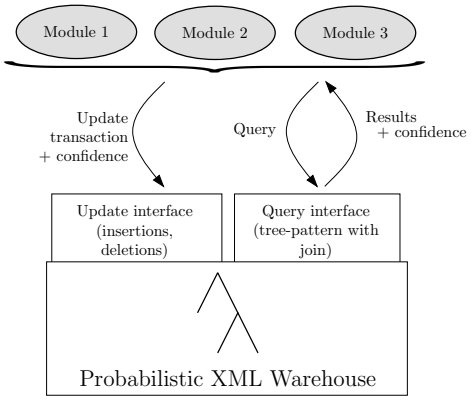

Fig. 1. Queries and Updates on a Probabilistic XML Warehouse

process, suggest the design of a *content-centric* system, as shown on Fig. 1 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.

3 Preliminary Definitions

In this section, we present preliminary definitions that are used in the 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 ε, 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 ordering of siblings.

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

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

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 *Ψ (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 Ψ *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* $Q(t')$.

We next define 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. $S \cap S' = \varnothing$. An *insertion* is an expression $i(t, n, t')$ where n is in S, the node where t' is to be inserted. A *deletion* is an expression $d(t, n)$ where n is in $S - \{r\}$. Node n is removed as well as all its descendants. 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 and* 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.

Definition 5. *Let* $\tau = (Q, U)$ *be an update transaction. Let* t *be a 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 τ *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* $D_1 \dots D_q$.

4 Two Simple Models

A natural way of representing probabilistic information is to list all *possible worlds*, each with its probability. See the example in Fig. 2. 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 a rather general, if not practical, way of representing probabilistic semi-structured information.

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

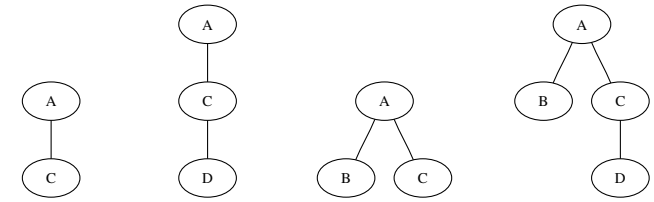

Fig. 2. Example Possible Worlds set

a PW set T is obtained by regrouping the identical component trees into one component and summing their probabilities. In the remaining of this paper, we will assume that all PW sets are normalized. 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. A *probabilistic update transaction* is a pair (τ, c) where τ is an update transaction and $c \in]0; 1]$ is the *confidence* we have in the transaction.

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

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

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.

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 more complex models.

In the remaining of this section, we discuss another natural model, namely the 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. The intuition is that it captures the probability of that node to be present assuming its parent is. A limitation of

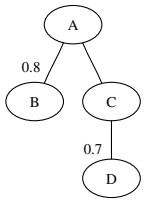

Fig. 3. Example SP tree

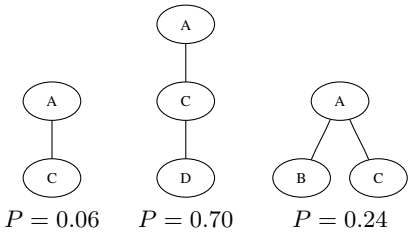

Fig. 4. More complex PW set

this model is that the only probability dependency captured is between the dependencies of nodes in a parent/child relationship.

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

Such an SP tree is represented as in Fig. 3.

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 $[\![T]\!]$, is defined as the normalization of the set $\{(t_X, p_X) \mid X \subseteq S\})$. 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. As an example, the semantics of the SP tree of Fig. 3 is the PW set on Fig. 2.

A natural question is then whether the SP tree model is as expressive as the PW set model. The answer is no. Figure 4 is an example of a PW set that has no equivalent SP tree. This negative result is not a sufficient reason for rejecting the SP model, since one might argue that PW sets not representable in the SP model are of little practical interest. The following result, however, demonstrates that the SP tree model does not meet basic requirements in terms of update, so motivates the model we introduce in the next section.

Proposition 1. *There exists an SP tree T and a probabilistic update transaction (τ, c) such that there is no SP tree whose semantics is $(\tau, c)([\![T]\!])$.*

In other words, "SP trees are not closed under updates". This comes from the fact that dependencies between nodes are not expressible 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. We next present a model that overcomes this limitation.

5 The 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.

The conditions we use are defined using the auxiliary concept of *probabilistic event*. Given a set W of event names, a probability distribution π 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. 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 10. *A fuzzy tree T is a 3-uple (t, π, γ) where $t = (S, E, r, \varphi, \nu)$ is a data tree, π is some probability distribution over some set W of events and γ assigns event conditions to nodes in $S - \{r\}$.*

Definition 11. *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 $[\![T]\!]$, defined as the normalization of:*

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

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 Fig. 2 and 4, respectively, for the fuzzy trees of Fig. 5. 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 Fig. 4. Actually, the following important result states that the fuzzy tree model is as expressive as the PW model.

Theorem 1. *For each PW set X, there exists a fuzzy tree T such that $X = [\![T]\!]$.*

We now define queries on fuzzy trees:

Definition 12. *The result of Q on a fuzzy tree $T = (t, \pi, \gamma)$, denoted $Q(T)$, is the normalization of $\bigcup_{u \in Q(t)} \left\{ \left(u, eval\left(\bigcup_{n \text{ node of } u} \gamma(n) \right) \right) \right\}$ 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 probabilities is 0, the element is removed. If the resulting set is empty, Q does not match T.

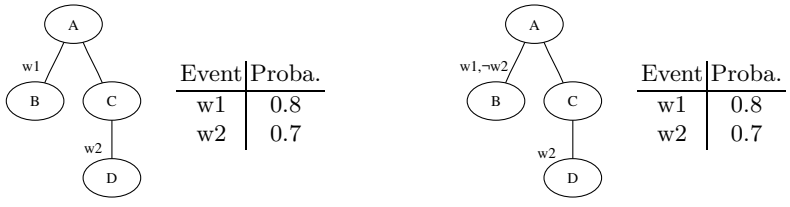

Fig. 5. Sample fuzzy trees

Theorem 2. *Let T be a fuzzy tree and Q be a TPWJ query. Q matches T if and only if Q matches $[\![T]\!]$. Moreover, $Q(T) = Q([\![T]\!])$.*

Finally, we show that unlike the SP model, the fuzzy tree model supports arbitrary probabilistic update transactions. Let (τ, 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.

Consider the case where $|Q(T)| = 1$, that is where the position of update operations is uniquely defined (the extension when $|Q(T)| > 1$ is straightforward and detailed in [6]). 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 to be applied to the inserted and deleted nodes. The result of (τ, c) on T is denoted $(\tau, c)(T)$, is the fuzzy tree obtained from t by applying insertions and deletions in the following way.

Insertions are performed at the position mapped by Q on t in u. If n is the position to insert a subtree t', and $cond_{\text{ancestors}}$ is the union of the event conditions on the (strict) ancestors of n, t' is inserted and its root is assigned the condition $\{w\} \cup (cond - (\gamma(n) \cup cond_{\text{ancestors}}))$.

Deletions are performed at the position mapped by Q on t. Let n be the node to be deleted and $cond_{\text{ancestors}}$ be the union of the event conditions on the (strict) ancestors of n. Let $cond_{new} = \{w\} \cup cond - (\gamma(n) \cup cond_{\text{ancestors}})$. 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 $\gamma(n) \cup \{a_1, \neg a_2\} \dots$ The last copy of n is annotated with conditions $\gamma(n) \cup \{a_1 \dots a_{n-1}, \neg a_n\}$.

Theorem 3. *Let (τ, c) be a probabilistic update transaction and T a fuzzy tree. Then $[\![(\tau, c)(T)]\!] = (\tau, c)([\![T]\!])$.*

The fuzzy tree model provides a concise representation of imprecision. Updates can be captured in the model. Simple updates (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. Complex updates may still be costly.

An interesting side benefit of using the fuzzy tree model is the possibility to keep *lineage* (or *provenance*) 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 of the corresponding transaction. Note that these variables are preserved throughout the whole process; a fuzzy tree system is 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.

6 Implementation

This section briefly discusses our implementation of a fuzzy tree system that 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 themselves are represented

as XML fragments. The query evaluation over fuzzy trees is implemented using the Qizx/open Java XQuery engine [8]. TPWJ queries are compiled into XQuery queries, whose results are converted to the minimal subtrees referred to in Definition 2. A post-processing step is performed on the resulting subtrees to compute the associated probabilities. For optimization, query processing relies on a dataguide of the document obtained by using the XML Summary Drawer described in [9]. Finally, updates are performed directly on the XML tree, following the rules described in Sect. 5.

7 Conclusion

The topic of probabilistic databases has been intensively studied, see for instance [4, 3, 10, 11, 12], and [2, 13] for more recent works. In [13], 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 data elements comes from the *conditional tables* of [4].

A relatively small number of works have dealt with the representation of probabilistic semi-structured data. In [14], Dekhtyar et al. use a semi-structured database to store complex probabilistic distributions of data which is essentially relational. Works closer to ours are [15, 16, 17]. Nierman et al. [15] describe a variant of the SP model and present strategies for efficient evaluations of logical queries. In [16], a complex model, based on directed acyclic graphs, is developed, along with an algebraic query language. Finally, Keulen et al. [17] 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, one main contribution of this paper.

The work presented here is part of a larger project on the construction of content warehouses from (Hidden) Web resources as described in Sect. 2. While working on this topic, we realized that imprecision has to be a core part of the XML-based warehouse since ad-hoc processing of imprecision simply does 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 with a real application.

A most important direction of research is to develop optimization techniques tailored to the fuzzy tree model. In particular, one would like to trim query evaluation branches that would provide data with too low confidence. Also, we want to study fuzzy tree simplification, i.e. finding more compact representations of imprecise data. 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. Actually, a similar construction can provide a representation of the answer as a fuzzy tree; the details are omitted. Finally, an interesting aspect is schema validation. Suppose we have a fuzzy tree representation T of some data, conforming to some DTD D. Its semantics can be seen as $X = \{[\![T]\!] \cap sat(D)\}$ 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. Abiteboul, S., Grahne, G.: Update semantics for incomplete databases. In: Very Large Data Bases, Stockholm, Sweden (1985)
6. Abiteboul, S., Senellart, P.: Querying and updating probabilistic information in XML. Technical Report 435, GEMO, Inria Futurs, Orsay, France (2005)
7. BrightPlanet: The Deep Web: Surfacing hidden value. White Paper (2000)
8. Franc, X.: Qizx/open (2005) `http://www.xfra.net/qizxopen/`.
9. Arion, A., Bonifati, A., Manolescu, I., Pugliese, A.: Path summaries and path partitioning in modern XML databases. Technical Report 437, Gemo (2005)
10. Cavallo, R., Pittarelli, M.: The theory of probabilistic databases. In: Very Large Data Bases. (1987) 71–81
11. Barbará, D., Garcia-Molina, H., Porter, D.: The management of probabilistic data. IEEE Transactions on Knowledge and Data Engineering **4** (1992) 487–502
12. Fuhr, N., Rölleke, T.: A probabilistic relational algebra for the integration of information retrieval and database systems. ACM Trans. Inf. Syst. **15** (1997)
13. 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)
14. Dekhtyar, A., Goldsmith, J., Hawkes, S.R.: Semistructured probabilistic databases. In: Statistical and Scientific Database Management, Tokyo, Japan (2001) 36–45
15. Nierman, A., Jagadish, H.V.: ProTDB: Probabilistic data in XML. In: Very Large Data Bases, Hong Kong, China (2002)
16. 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
17. van Keulen, M., de Keijzer, A., Alink, W.: A probabilistic XML approach to data integration. In: International Conference on Data Engineering. (2005) 459–470