Modeling and Querying Probabilistic XML Data

Benny Kimelfeld
IBM Almaden Research Center
kimelfeld@us.ibm.com

*Yehoshua Sagiv**
The Hebrew University of Jerusalem
sagiv@cs.huji.ac.il

1 Introduction

We survey recent results on modeling and querying probabilistic XML data. The literature contains a plethora of probabilistic XML models [2, 13, 14, 18, 21, 24, 27], and most of them can be represented by means of *p-documents* [18] that have, in addition to ordinary nodes, *distributional* nodes that specify the probabilistic process of generating a random document. The above models are families of p-documents that differ in the *types* of distributional nodes in use.

The focus of this survey is on the tradeoff between the ability to express real-world probabilistic data (in particular, by taking correlations between atomic events into account) and the efficiency of query evaluation. We concentrate on two important issues. The first is the ability to efficiently translate a p-document of one family into that of another. The second is the complexity of query evaluation over p-documents (under the usual semantics of querying probabilistic data, e.g., [4, 9, 10]). It turns out that efficient evaluation of a large class of queries (i.e., twig patterns with projection and aggregate functions) is realizable in models where distributional nodes are probabilistically independent. In other models, the evaluation of a query with projection is very often intractable. In comparison, very simple conjunctive queries are intractable over probabilistic models of relational databases, even when the tuples are probabilistically independent [9, 10].

To handle the limitation exhibited by the above tradeoff, various approaches have been proposed. The first is to allow query answers to be *approximate* [18], which makes the evaluation of twig patterns with projection tractable in the most expressive family among those considered. This tractability, however, does not carry over to non-monotonic queries, such as twig patterns with negation or aggregation. The approach presented in [7] combines the assumption about the independence of distributional nodes with the assertion of (a fixed set of) *constraints*, and the result is a model that is capable of representing complex correlations between atomic events, but not at the expense of efficiency.

*The work of this author was supported by The Israel Science Foundation (Grant 893/05).

2 Probabilistic XML

We model XML documents as unranked and unordered trees. Each node v has a *label* and a unique *identifier* that are denoted by $\lambda(v)$ and $id(v)$, respectively. When representing an XML document as a tree, a node corresponds to either an *element*, an *attribute* or a *value* (i.e., PCDATA or the value of an attribute). Accordingly, the label of a node corresponds to either an element name (i.e., tag), an attribute name or a value. Trees of the above form are called *documents*.

A *probabilistic XML space* (abbr. *px-space*) $\tilde{\mathscr{D}}$ is a pair (Ω, p), where Ω is a nonempty and finite set of documents, and $p : \Omega \to \mathbb{Q}^+$ maps every document $d \in \Omega$ to a positive rational number $p(d)$, such that $\sum_{d\in\Omega} p(d) = 1$. The set Ω is the *sample space* of $\tilde{\mathscr{D}}$, and p is the *probability distribution*. The documents of Ω are also called *possible worlds* (or *samples*). We identify $\tilde{\mathscr{D}}$ with its sample space Ω; for example, we write $d \in \tilde{\mathscr{D}}$ instead of $d \in \Omega$.

Typically, a px-space describes uncertainty in many parts of the data. Hence, its sample space may be too large for allowing an explicit representation. Next, we describe a compact representation of a px-space $\tilde{\mathscr{D}}$ by means of a p-document, which is (a description of) a probabilistic process that generates a random document $d \in \tilde{\mathscr{D}}$ with probability $p(d)$.

2.1 The P-Document Model

A *p-document* is a tree $\tilde{\mathscr{P}}$ that consists of two types of nodes. *Ordinary* nodes are those described at the beginning of Section 2, namely, each one has a label and a unique identifier. Ordinary nodes may appear in the documents of the sample space. *Distributional* nodes, on the other hand, are only used for defining the probabilistic process that generates random documents (but they do not actually occur in those documents). In Section 3, several types of distributional nodes are defined. For now, it is sufficient to realize that each distributional node v has a probability

distribution over subsets of its children. In the probabilistic process that generates a random document, v randomly chooses a subset of its children according to the distribution specified for v. In a p-document, the root and leaves are ordinary nodes.

As an example, Figure 1 shows a p-document $\tilde{\mathscr{P}}$. Each ordinary node v is represented by the string $id(x).\lambda(v)$ (e.g., "3.member"). Distributional nodes are depicted as rounded-corner rectangles. The type of a distributional node is indicated by the string (e.g., ind or mux) appearing inside the rectangle. These types are discussed in Section 3.

Given a p-document $\tilde{\mathscr{P}}$, a random document is generated in two steps. First, each distributional node randomly chooses a subset of its children. Note that the choices of different nodes are not necessarily probabilistically independent. All the unchosen children and their descendants (even descendants that have been chosen by their own parents) are deleted. The second step removes all the distributional nodes. If an ordinary node u remains, but its parent is removed, then the new parent of u is the lowest ordinary node v of $\tilde{\mathscr{P}}$, such that v is a proper ancestor of u. Note that when distributional nodes have distributional children, it may happen that the same document is obtained from two different applications of the first step. For additional details, see [18, 19].

3 Concrete Models

3.1 Types of Distributional Nodes

To obtain a concrete p-document, every distributional node should have a specific probability distribution of choosing a subset of its children. We define five types of distributional nodes, each one is characterized by a different way of describing that probability distribution. A node v of type ind specifies for each child w, the probability of choosing w. This probability is independent of the other choices of children. As a special case, a node v of type det always (deterministically) chooses all of its children. If the type of v is mux, then choices of different children are mutually exclusive. That is, v chooses at most one of its children, and it specifies the probability of choosing each child (so the sum of these probabilities is at most 1). A node v of type exp specifies the probability distribution explicitly. That is, v lists subsets of children and their probabilities of being chosen. We assume that the probabilities specified in a p-document are all non-zero, because it is useless to consider choices of children that have a zero probability of occurring.

In the above four types, the underlying assumption is that the choices of different distributional nodes are probabilistically independent. The next type makes it possible to introduce correlations between choices of nodes. When distributional nodes of type cie appear in a p-document $\tilde{\mathscr{P}}$, it means that $\tilde{\mathscr{P}}$ has independent random Boolean variables $e_1, \ldots, e_m$, called *event variables*. For each variable e_i, the p-document $\tilde{\mathscr{P}}$ specifies the probability $p(e_i)$ that e_i is **true**. Each node v of type cie specifies for every child w, a conjunction[1] $\alpha^v(w) = a_1 \wedge \cdots \wedge a_{l_w}$, where each a_j is either e_i or $\neg e_i$ for some $1 \leq i \leq m$ (note that different cie nodes can share common event variables). When generating a random document, values for $e_1, \ldots, e_m$ are randomly picked out, and a child is chosen if its corresponding conjunction is satisfied.

Next, we discuss how to compute the probability of a possible world d that belongs to the px-space defined by a p-document $\tilde{\mathscr{P}}$. Consider an execution of the probabilistic process that generates the subtree s of $\tilde{\mathscr{P}}$ in the first step and then produces d in the second step. For each distributional node v of s, such that the type of v is not cie, let p_v be the probability that v chooses exactly the children that it has in s. It is easy to compute p_v from the probability distribution specified for v. For the cie nodes, we have to compute the probability p_e of all the truth assignments τ to the event variables, such that for every cie node v of s the following holds. For all children w of v, if w appears in s, then τ satisfies $\alpha^v(w)$; otherwise, it does not. Let $p(s)$ be the product of p_e and all the p_v. Note that $p(s)$ is the probability that each distributional node of s chooses *exactly* the children that it has in s. Equivalently, it is the probability of getting s at the end of the first step. The probability of the possible world d is the sum of the probabilities $p(s)$ over all the subtrees s that yield d at the end of the second step. Note that computing each of p_e, $p(s)$ and the probability of d is generally intractable. But, if there are no cie nodes in $\tilde{\mathscr{P}}$, then the three probabilities can be computed efficiently.

3.2 Families of P-Documents

We denote by $\mathsf{PrXML}^{\{\text{type}_1,\text{type}_2,\ldots\}}$ the family of all the p-documents, such that the types of their distributional nodes are among those listed in the superscript. For example, the p-documents of $\mathsf{PrXML}^{\{\mathsf{ind},\mathsf{mux}\}}$ use only ind and mux nodes.

The simple case of using a distributional node is when both its parent and children are ordinary. Then, the role of the distributional node is to choose ordinary children for its ordinary parent. Sometimes, however, we can obtain more complex probability distributions (over the space of documents) by constructing hierarchies of distributional nodes.

Formally, a p-document $\tilde{\mathscr{P}}$ is *distributional-*

[1] We assume that the conjunction is satisfiable, that is, it does not include an event variable and its negation.

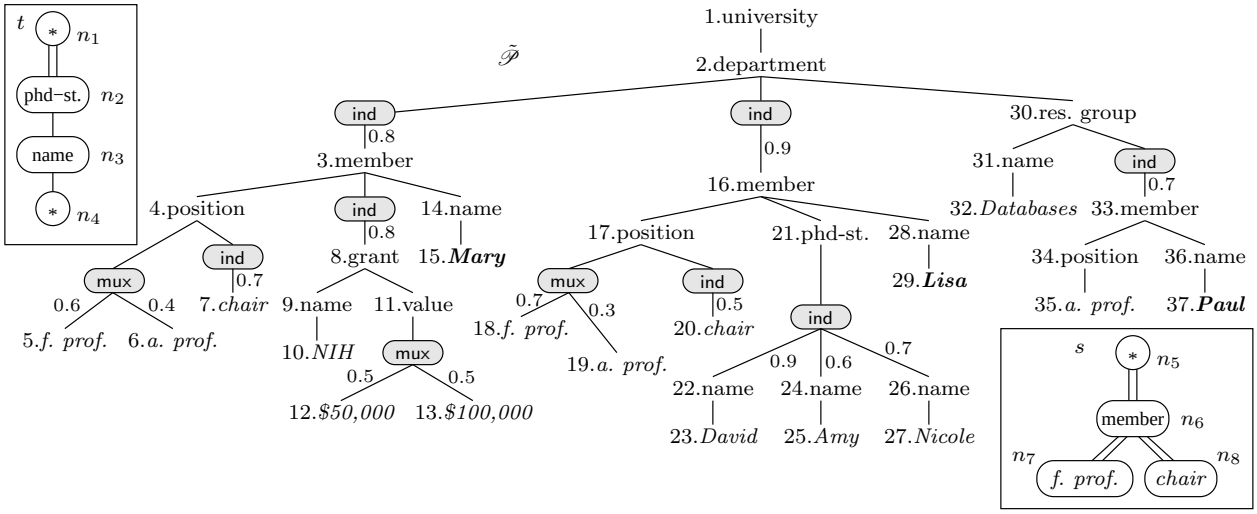

Figure 1: A p-document $\tilde{\mathscr{P}}$ and twigs t and s

hierarchy free (abbr. DHF) if every distributional node of $\tilde{\mathscr{P}}$ has only ordinary children. As an example, the p-document $\tilde{\mathscr{P}}$ of Figure 1 is DHF. If $\mathscr{F}$ is a set of p-documents, then $\mathscr{F}_{|\not\curlywedge}$ denotes the restriction of $\mathscr{F}$ to its DHF p-documents.

3.3 Expressiveness of Models

We have five types of distributional nodes and the option of either allowing or forbidding hierarchies. It gives more than fifty combinations—do all of them create families of p-documents that are inherently different from one another? In this section, we compare different families in terms of their expressiveness, using the notion of translations. A family $\mathscr{F}_1$ can be *translated* into a family $\mathscr{F}_2$ if there is an algorithm that accepts as input a p-document $\tilde{\mathscr{P}}_1$ of $\mathscr{F}_1$ and constructs as output a $\tilde{\mathscr{P}}_2$ of $\mathscr{F}_2$, such that $\tilde{\mathscr{P}}_1$ and $\tilde{\mathscr{P}}_2$ describe the same px-space. The translation is *efficient* if the algorithm runs in polynomial time.

As a simple example, we compare the families $\mathsf{PrXML}^{\{\mathsf{ind}\}}$ and $\mathsf{PrXML}^{\{\mathsf{mux}\}}$. There is no translation of $\mathsf{PrXML}^{\{\mathsf{mux}\}}$ into $\mathsf{PrXML}^{\{\mathsf{ind}\}}$, because ind nodes cannot express mutually exclusive choices. In particular, consider the p-document $\tilde{\mathscr{P}}_1$ of Figure 2(a). Note that rectangles and circles are distributional and ordinary nodes, respectively; furthermore, for each child of a distributional node, the probability of choosing it is written next to its incoming edge. $\tilde{\mathscr{P}}_1$ is in $\mathsf{PrXML}^{\{\mathsf{mux}\}}$ and it creates exactly two possible worlds that have the sets of nodes $\{v, u_1\}$ and $\{v, u_2\}$. No p-document of $\mathsf{PrXML}^{\{\mathsf{ind}\}}$ can yield these two possible worlds without also generating a fourth one that has the set of nodes $\{v, u_1, u_2\}$.

Figure 4 shows how to translate each ind node in a p-document of $\mathsf{PrXML}^{\{\mathsf{ind}\}}_{|\not\curlywedge}$ to several mux nodes (note that the probability of each w_i remains the same). Hence, $\mathsf{PrXML}^{\{\mathsf{ind}\}}_{|\not\curlywedge}$ is efficiently translatable into $\mathsf{PrXML}^{\{\mathsf{mux}\}}$.

However, $\mathsf{PrXML}^{\{\mathsf{ind}\}}$ cannot be translated into $\mathsf{PrXML}^{\{\mathsf{mux}\}}$. To see why, consider the p-document $\tilde{\mathscr{P}}_2 \in \mathsf{PrXML}^{\{\mathsf{ind}\}}$ of Figure 2(b). We define $E(x)$ as the event "node x appears in some possible world of $\tilde{\mathscr{P}}_2$." The events $E(u_1)$ and $E(u_2)$ have the same

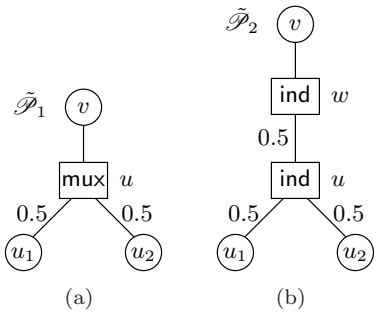

Figure 2: P-documents: (a) $\tilde{\mathscr{P}}_1$ and (b) $\tilde{\mathscr{P}}_2$

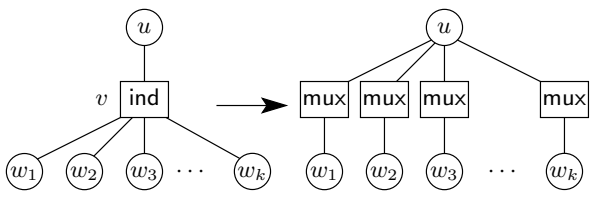

Figure 4: Translating $\mathsf{PrXML}^{\{\mathsf{ind}\}}_{|\not\curlywedge}$ to $\mathsf{PrXML}^{\{\mathsf{mux}\}}$

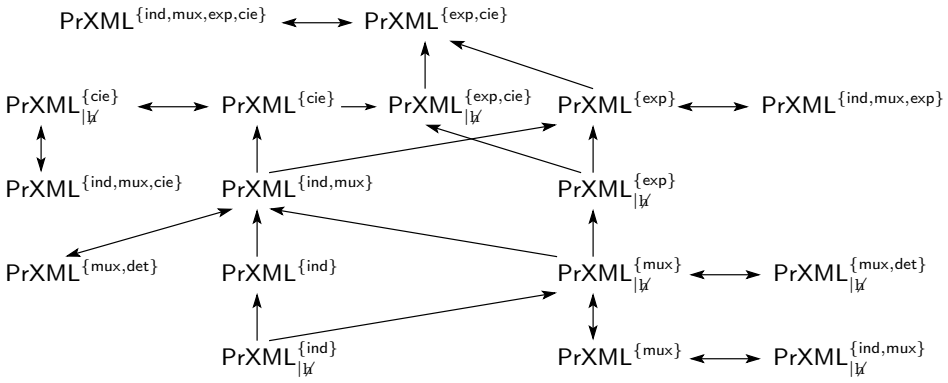

Figure 3: Efficient translations between families of p-documents

prior probability, namely, 0.25. But the conditional probability of $E(u_1)$, given the occurrence of $E(u_2)$, is 0.5 (because if u_2 appears in a possible world, it implies that node u has been chosen by its parent w). Suppose that some p-document $\tilde{\mathscr{P}}' \in \mathsf{PrXML}^{\{\mathsf{mux}\}}$ generates the same px-space as $\tilde{\mathscr{P}}_2$. A simple case analysis shows that in the px-space defined by $\tilde{\mathscr{P}}'$, the events $E(u_1)$ and $E(u_2)$ are either mutually exclusive or independent. Hence, no p-document of $\mathsf{PrXML}^{\{\mathsf{mux}\}}$ can generate the same px-space as $\tilde{\mathscr{P}}_2$.

Thus, the families $\mathsf{PrXML}^{\{\mathsf{ind}\}}$ and $\mathsf{PrXML}^{\{\mathsf{mux}\}}$ are incomparable in terms of expressive power. Note that both families are subsets of $\mathsf{PrXML}^{\{\mathsf{ind,mux}\}}$. Interestingly, the family $\mathsf{PrXML}^{\{\mathsf{ind,mux}\}}$ is efficiently translatable into $\mathsf{PrXML}^{\{\mathsf{mux,det}\}}$, as illustrated in Figure 5. The converse is trivial, because a det node is a special case of an ind node.

A thorough study of translations between families of p-documents is done in [1]. The results are summarized in Figure 3. An arrow means that there is an efficient translation in the specified direction. The figure is complete in the sense that if there is no directed path from a family $\mathscr{F}_1$ to another family $\mathscr{F}_2$, then an efficient translation does not exist.

3.4 Object and Value Semantics

Thus far, we have used the *object-based semantics*, namely, two documents cannot be the same if one has a node id that does not appear in the other. Sometimes, node id's are not important per se and, hence, the *value-based semantics* might be more suitable. Under this semantics, two documents are deemed the same if they are isomorphic. Formally, d_1 and d_2 are *isomorphic* if there is a one-to-one correspondence h between the nodes of d_1 and those of d_2, such that h preserves the tree structure and the labels, but not necessarily the id's.

When working with the value-based semantics, the intrinsic measure of uncertainty associated with a document d is the probability that a random possible world is isomorphic to d. In comparison, under the object-based semantics, we are interested in the probability that a random possible world is d itself. This distinction gives rise to the notion of a *v-translation* that transforms a p-document $\tilde{\mathscr{P}}_1$ to $\tilde{\mathscr{P}}_2$, such that the two generate isomorphic px-spaces. In the previous section, we actually discussed *o-translations* that are founded on the object-based semantics. Note that an o-translation is also a v-translation, but the converse is not necessarily true. Therefore, the existence of a directed path in Figure 3 means that there is an efficient v-translation. It is not known whether Figure 3 is complete for efficient v-translations. Notwithstanding, [1] shows that in many cases, the lack of a directed path indicates that there is no efficient v-translation. The main open problem is the following: Is $\mathsf{PrXML}^{\{\mathsf{exp}\}}$ efficiently v-translatable to $\mathsf{PrXML}^{\{\mathsf{mux,det}\}}$, or at least to $\mathsf{PrXML}^{\{\mathsf{cie}\}}$?

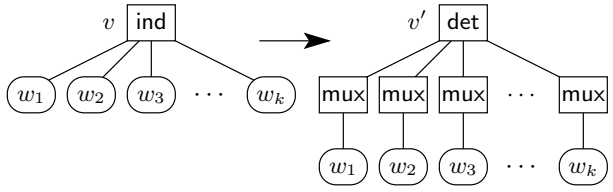

Figure 5: Translating $\mathsf{PrXML}^{\{\mathsf{ind,mux}\}}$ to $\mathsf{PrXML}^{\{\mathsf{mux,det}\}}$

3.5 Previously Studied Models

The family $\mathsf{PrXML}^{\{\mathsf{ind,mux}\}}$ is the ProTDB model of [21]. This model has also been studied in [7, 19]. The probabilistic XML model[2] of [27] is a subset of

[2]In the probabilistic documents of [27], the root is distributional. We can assume that a dummy ordinary node is added for compliance with the definition of p-documents.

$\mathsf{PrXML}^{\{\mathsf{mux},\mathsf{det}\}}$, where mux nodes (called "probability nodes") have only det nodes as children (called "possibility nodes") and det nodes have only ordinary children (called "XML nodes").

The model of probabilistic XML that was investigated in [2, 24] is $\mathsf{PrXML}^{\{\mathsf{cie}\}}$. The "simple probabilistic trees" of [2] are actually the family $\mathsf{PrXML}^{\{\mathsf{ind}\}}_{\not\Downarrow}$ (hierarchies make a difference in this case).

The work of [14] introduced a model of probabilistic XML graphs, where each node explicitly specifies the probability distribution over its possible sets of children. Restricting their XML graphs to trees yields a sub-family of $\mathsf{PrXML}^{\{\mathsf{exp}\}}_{\not\Downarrow}$ (a lack of hierarchies is significant when only exp nodes are allowed). The same is true for [13] if we restrict their intervals to points.

4 Querying Probabilistic XML

In this section, we survey results on query evaluation over probabilistic XML. The focus is on queries that are based on twig patterns [3, 5]. Formally, a *twig* is a tree t with *child* and *descendant* edges, which are depicted by single and double lines, respectively, in the rectangular boxes of Figure 1. A *match* of a twig t in a document d is a mapping μ from the nodes of t to those of d, such that μ maps root to root, nodes to nodes, child edges to edges, and descendant edges to paths (with at least one edge). In addition, each node n of the twig has a unary condition $c_n(\cdot)$, and $\mu(n)$ must satisfy this condition, namely, $c_n(\mu(n))$ should evaluate to **true**. The simplest conditions are $\lambda(\mu(n)) = l$ (i.e., the label of $\mu(n)$ is l) and **true**; the latter is also known as the *wildcard* and denoted by the symbol $*$. In the twig t of Figure 1, these simple conditions appear as a label or a $*$ inside each node.

The conventional semantics of evaluating a twig t (or any other type of query) is to find the matches of t and their probabilities (e.g., [9, 10]). That is, we need to compute a function p over all the mappings μ, such that $p(\mu)$ is the probability that μ is a match of t in a random possible world. The set of answers comprises all matches μ, such that $p(\mu) > 0$. Given a twig t, a p-document $\tilde{\mathscr{P}}$ and a mapping μ, the probability $p(\mu)$ can be computed efficiently, even if there are cie nodes, based on the following observation. Let s be the minimal subtree of $\tilde{\mathscr{P}}$ that includes the nodes in the image of μ. Observe that $p(\mu)$ is the same as the probability that each distributional node of s chooses *at least* the children that it has in s. (In comparison, the probability $p(s)$ defined at the end of Section 3.1 is that of choosing *exactly* the children that appear in s.) For example, consider the twig t and the p-document $\tilde{\mathscr{P}}$ of Figure 1. Let μ be the match defined by $\mu(n_1) = 1$, $\mu(n_2) = 21$, $\mu(n_3) = 26$ and $\mu(n_4) = 27$. The probability of μ is $0.9 \cdot 0.7 = 0.63$.

However, computing a query that involves projection is not so easy. For example, there are extremely simple conjunctive queries that are intractable over probabilistic relational databases, even if choices of distinct tuples are assumed to be independent [9, 10]. The ultimate usage of projection is to apply the Boolean interpretation. In the case of a twig t and an ordinary document d, it means to determine whether d *satisfies* t, denoted by $d \models t$; that is, to decide whether there is a match of t in d.

For a given p-document $\tilde{\mathscr{P}}$, we use $\mathscr{P}$ (i.e., without the tilde sign) to denote the random variable that represents a possible world of $\tilde{\mathscr{P}}$. Evaluating a Boolean twig t over $\tilde{\mathscr{P}}$ amounts to computing $\Pr(\mathscr{P} \models t)$, which is the probability that a random possible world of $\tilde{\mathscr{P}}$ satisfies t. By definition, $\Pr(\mathscr{P} \models t)$ is the sum of probabilities of all possible worlds d, such that d satisfies t.

Consider, for example, the p-document $\tilde{\mathscr{P}}$ and the twig t of Figure 1. A possible world of $\tilde{\mathscr{P}}$ satisfies t if it contains either node 23, node 25 or node 27. For each of these three nodes alone, we can compute the probability that it appears in a random possible world, as explained above. But the sum of these three probabilities is not what we are looking for, because these are not disjoint events (i.e., some possible worlds include all three nodes while others include only one or two of them).

In general, evaluating Boolean twigs over p-documents is a hard problem. In [18], it is shown that every nontrivial Boolean twig has an intractable data complexity over p-documents of $\mathsf{PrXML}^{\{\mathsf{cie}\}}$. By definition, a Boolean twig is *trivial* if it has only one node (i.e., it is a condition on the root of the document) or it contains a node with an unsatisfiable condition (i.e., equivalent to **false**). Recall that $\mathrm{FP}^{\#\mathrm{P}}$ is the class of functions that are efficiently computable using an oracle to some function in[3] #P.

Theorem 4.1 [18] *The evaluation of every nontrivial Boolean twig over* $\mathsf{PrXML}^{\{\mathsf{cie}\}}$ *is* $\mathrm{FP}^{\#\mathrm{P}}$*-complete.*

In contrast to Theorem 4.1, [19] shows that over $\mathsf{PrXML}^{\{\mathsf{ind},\mathsf{mux}\}}$ (i.e., the ProTDB model of [21]), every Boolean twig can be efficiently evaluated under data complexity. In [18], this result is generalized to the family $\mathsf{PrXML}^{\{\mathsf{exp}\}}$. Thus, from Figure 3 and Theorem 4.1, it follows that the family $\mathsf{PrXML}^{\{\mathsf{exp}\}}$ is the maximal one, among those considered in the

[3] The class #P [26] is that of the functions that count the number of accepting paths of the input of an NP machine. By using an oracle to a #P-hard (or $\mathrm{FP}^{\#\mathrm{P}}$-hard) function, one can efficiently solve the entire polynomial hierarchy [25].

previous section, that allows efficient evaluation of Boolean twigs.

In [7], it is shown that tractable data complexity carries over to *c-formulae*, which are rather complex queries with aggregate functions. In particular, c-formulae are obtained by mutually nesting twigs and comparisons involving aggregate functions. A simple example of an atomic c-formula is $(\mathsf{count}(s)\ \theta\ R)$, where s is a *selector*, R is a rational number and θ is one of the operators $<, >, \leq, \geq, =$ and $\neq$. The selector s is a twig that computes a set of node ids, when given a document as input. The aggregate function count is applied to the set computed by s and the result is compared with R.

The aggregate functions min and max are also allowed in c-formulae, provided that labels are interpreted as numeric values. Another aggregate function is ratio. The simple atomic c-formulae that use ratio have the form $(\mathsf{ratio}(s,t)\ \theta\ R)$, where s, θ and R are as above, and t is a Boolean twig. The function $\mathsf{ratio}(s,t)$ is interpreted in a given document d as the ratio $|U|/|S|$, where S is the set of nodes that are selected by s, and U is the subset of S comprising all the nodes u, such that the subtree of d that is rooted at u satisfies t.

Theorem 4.2 [7] *Let q^A be a c-formula that uses the aggregate functions* count, ratio, min *and* max. *The evaluation of q^A over $\mathsf{PrXML}^{\{\mathsf{exp}\}}$ is in polynomial time.*[4]

It is also shown in [7] that Theorem 4.2 does not generalize to the aggregate functions sum and avg. For instance, it is intractable to compute the probability that the total sum (or the total average) of all the numeric labels in a random possible world is zero; in fact, it cannot even be efficiently approximated unless[5] NP=RP.

In comparison to Theorem 4.2, the evaluation algorithms of [18, 19] apply only to twigs (which are a subclass of c-formulae), but they are more efficient. In particular, the algorithms of [18, 19] are *fixed-parameter tractable*[6] [11, 22], whereas that of [7] is not.

Finally, in [7, 18, 19], it is shown how to apply their results to non-Boolean queries, namely, projection can be applied (to the type of queries they consider), but not necessarily in a total manner.

4.1 Approximate Query Evaluation

Let t be a twig. In the context of evaluating t over a p-document, a *fully polynomial randomized approximation scheme* (*FPRAS*) for t is a randomized algorithm A that, given a p-document $\tilde{\mathscr{P}}$ and an $\epsilon > 0$, returns a number $A(\tilde{\mathscr{P}}, \epsilon)$, such that[7]

$$\Pr\left((1-\epsilon)p \leq A(\tilde{\mathscr{P}}, \epsilon) \leq (1+\epsilon)p\right) \geq \frac{3}{4},$$

where $p = \Pr(\mathscr{P} \models t)$. Moreover, the running time of A is polynomial in $\tilde{\mathscr{P}}$ and in $1/\epsilon$.

In [18], it is shown that by using the Monte-Carlo approximation technique of [17] (similarly to the way it is done in [10]), twigs can be efficiently approximated over the maximal family considered in Figure 3.

Theorem 4.3 [18] *Every twig has an FPRAS over $\mathsf{PrXML}^{\{\mathsf{exp,cie}\}}$.*

By combining a simple twig and any aggregate function (among those considered above), it is possible to get a query that cannot be efficiently approximated over $\mathsf{PrXML}^{\{\mathsf{cie}\}}$ (for all $\epsilon > 0$), unless NP=RP. This is proved rather easily by using the following observation. It is NP-hard to test, for a given p-document $\tilde{\mathscr{P}} \in \mathsf{PrXML}^{\{\mathsf{cie}\}}$, whether the probability that $\tilde{\mathscr{P}}$ does *not* satisfy the twig a/b is nonzero.

4.2 Enumerating Matches of Twigs

By Theorem 4.2, twig queries can be efficiently evaluated under data complexity. Under *query-and-data* complexity (i.e., when both the p-document and the query are given as input), this result no longer holds. Moreover, it is already the case if we are restricted to the family $\mathsf{PrXML}^{\{\mathsf{mux}\}}$ and twig queries without descendant edges and projection [18]. This is explained by the NP-completeness of the problem of determining whether there is a nonzero probability that a match of a given twig in a given p-document of $\mathsf{PrXML}^{\{\mathsf{mux}\}}$ exists.

Note that for projection-free twig queries, the goal is essentially to enumerate all the matches with a nonzero probability, and determine the probability of each. Under query-and-data complexity, the number of matches can be exponential in the size of the input. Consequently, "polynomial time in the size of the input" is not a suitable yardstick. Instead,

[4] Similarly to [23], the numerical operands of the query q^A are not assumed to be fixed; rather, they are given as part of the input.

[5] Note that NP=RP implies that the whole polynomial hierarchy is recognizable by an efficient randomized algorithm with a bounded two-sided error (BPP) [28].

[6] In the function that gives the running time, the size of the query effects only the constant, but not the degree of the polynomial.

[7] Note that the choice of the reliability factor 3/4 is arbitrary, since for a given $\delta > 0$, one can enhance the reliability to $(1-\delta)$ by taking the median of $O(\log \delta)$ trials [15].

other yardsticks of efficiency are used in the literature. The common one is *polynomial total time*, namely, the running time is polynomial in the combined size of the input and the output. A stronger notion is that of *enumeration in incremental polynomial time* [16], namely, the ith answer is generated in time that is polynomial in the size of the input and that of the previous $i-1$ answers. The above NP-completeness result of [18] implies that one cannot efficiently enumerate all the matches μ of a given twig t in a p-document $\tilde{\mathscr{P}}$, such that μ has a nonzero probability. Rather surprisingly, [19] shows that this task (and even a generalized one) can be done efficiently if *maximal* matches (rather than the ordinary *complete* matches) are allowed.

Formally, a *partial match* of a twig t in a document d is a match of a *root-subtree* t' of t (i.e., t' is a subtree of t that contains the root of t) in d. In [19], the following problem was considered. Given a twig pattern t, a p-document $\tilde{\mathscr{P}}$ and a threshold $p \in [0,1]$, enumerate all the *maximal matches w.r.t. p*, namely, all the partial matches μ such that *(1)* the probability of μ is at least p, *and (2)* no partial match μ' with a probability at least p subsumes μ.

As an example, consider the p-document $\tilde{\mathscr{P}}$ and the twig s of Figure 1. Let $p = 0.4$ and μ_1, μ_2 and μ_3 be the partial matches defined as follows.

- $\mu_1(n_5) = 1$, $\mu_1(n_6) = 3$.
- $\mu_2(n_5) = 1$, $\mu_2(n_6) = 3$, $\mu_2(n_7) = 5$.
- $\mu_3(n_5) = 1$, $\mu_3(n_6) = 3$, $\mu_3(n_7) = 5$, $\mu_3(n_8) = 7$.

Note that μ_1 is subsumed by μ_2 and both μ_1 and μ_2 are subsumed by μ_3. The partial matches μ_1, μ_2 and μ_3 have the probabilities 0.8, $0.8 \cdot 0.6 = 0.48$ and $0.8 \cdot 0.6 \cdot 0.7 = 0.336$, respectively. Hence, μ_1 is not maximal w.r.t. p since it is subsumed by μ_2 that has a probability above p, and μ_2 is maximal since it is only subsumed by μ_3 that has a probability smaller than p. The result of [19] mentioned above is the following.

Theorem 4.4 [19] *Over* $\mathsf{PrXML}^{\{\mathsf{ind},\mathsf{mux}\}}$*, all maximal matches of a given twig pattern in a given p-document w.r.t. a given threshold can be enumerated in incremental polynomial time.*

Whether Theorem 4.4 can be generalized to the families $\mathsf{PrXML}^{\{\mathsf{exp}\}}$ or $\mathsf{PrXML}^{\{\mathsf{cie}\}}$ is yet unknown. The operation of *maximally joining probabilistic relations* [20] is essentially the probabilistic version of the *full disjunction* [12]. This operation is very similar to that of enumerating the maximal matches of a twig in a p-document, but its general case is intractable [20] (whereas full disjunctions are computable with polynomial delay [6]).

5 Asserting Constraints

The type cie of distributional nodes is unique in its ability to represent correlations between choices of children at different distributional nodes. However, as discussed in the previous section, this type extremely limits the ability to evaluate queries that involve projection. In contrast, the other types (namely, exp, ind, mux and det) allow efficient evaluation of highly expressive queries. But they also entail an inherent assumption of probabilistic independence. This assumption often severely limits the ability to model real-life data.

As an example, consider again the document $\tilde{\mathscr{P}}$ of Figure 1. This p-document can be thought of as a fragment of a probabilistic database that represents the result of screen scraping a university Web site. In particular, each probability embodies the degree of certainty, in making some specific choice, as determined during the screen-scraping process. In addition to these probabilities, available information about the university can imply intricate correlations between different choices. For example, suppose that the following facts are known to hold.

1. Every department with three or more members has a chair (and there is at most one chair).
2. A chair must be a full professor.
3. At least 95% of the associate professors have at most two Ph.D. students.

This information implies that many possible worlds of $\tilde{\mathscr{P}}$ are actually inconceivable. Moreover, rather intricate correlations exist between the entities of $\tilde{\mathscr{P}}$. For instance, given that Lisa is a faculty member, there are three individuals that can be chosen as her students. These choices, however, are not probabilistically independent, because if Lisa is an associate professor, then it is most likely that she has no more than two students. Moreover, the second fact implies that these choices also depend on whether Lisa is the chair. Lastly, the first two facts imply that the academic rank of Lisa probabilistically depends on the likelihood that Mary is the chair, and also on whether Paul is a member of the department (because if there are three members, then there is higher chance that Lisa is the chair and, hence, a full professor).

One way of incorporating the available information is to use cie nodes and correlate them by means of shared event variables. However, it is not clear how (and whether) this can be done efficiently (i.e., the resulting p-document should not be too large). Moreover, even if that could be done efficiently, using

cie nodes has its own limitations (as discussed earlier). A similar problem exists in other known models, such as those based on *Bayesian networks* (where approximating the probability of simple events is intractable [8]). A direct and convenient approach is proposed in [7], namely, representing the probability space of possible worlds by means of a *PXDB*.

A *PXDB* is a px-space that is represented by a pair $(\tilde{\mathscr{P}}, \mathscr{C})$, where $\tilde{\mathscr{P}}$ is a p-document of $\mathsf{PrXML}^{\{\mathsf{exp}\}}$ (which, by Figure 3, effectively allows all the types of distributional nodes, except for cie) and $\mathscr{C}$ is a set of *constraints*, such as the above three facts. The px-space $\tilde{\mathscr{D}}$ that is given by $(\tilde{\mathscr{P}}, \mathscr{C})$ is *well defined* if there is a nonzero probability that a random document of $\tilde{\mathscr{P}}$ satisfies $\mathscr{C}$. In that case, $\tilde{\mathscr{D}}$ is the subspace of $\tilde{\mathscr{P}}$ that comprises all the possible worlds that satisfy each of the constraints (and, as usual, $\mathscr{D}$ is the random variable associated with $\tilde{\mathscr{D}}$). In particular, for a document d that satisfies $\mathscr{C}$ (denoted by $d \models \mathscr{C}$), the probability $\Pr(\mathscr{D} = d)$ is given by

$$\Pr(\mathscr{D} = d) = \Pr(\mathscr{P} = d \mid \mathscr{P} \models \mathscr{C}) = \frac{\Pr(\mathscr{P} = d)}{\Pr(\mathscr{P} \models \mathscr{C})}\,.$$

The above three facts about the university of Figure 1 can be easily expressed as c-formulae that use the aggregate functions count and ratio (recall that c-formulae were discussed in Section 4). The importance of this observation lies in Theorem 4.2. In particular, it is shown in [7] how Theorem 4.2 can be used for obtaining the following result.

Theorem 5.1 [7] *Let $\mathscr{C}$ be a fixed set of c-formulae that use the aggregate functions* count, ratio, min *and* max. *The following three tasks can be performed efficiently, given a PXDB $\tilde{\mathscr{D}} = (\tilde{\mathscr{P}}, \mathscr{C})$.*

- *Testing well-definedness of $\tilde{\mathscr{D}}$.*
- *Evaluating a twig (or a c-formulae with the above aggregate functions) over $\tilde{\mathscr{D}}$.*
- *Sampling $\tilde{\mathscr{D}}$.*

Sampling a PXDB $\tilde{\mathscr{D}}$ is the task of emulating $\tilde{\mathscr{D}}$ by randomly generating a document of $\tilde{\mathscr{D}}$, such that the probability of generating each document d is equal to $\Pr(\mathscr{D} = d)$.

Observe that Theorem 5.1 makes the limiting (yet necessary) assumption that the set $\mathscr{C}$ of constraints is fixed (although the numerical values that appear in $\mathscr{C}$ are given as part of the input). Therefore, one can effectively utilize this result when the correlations between the represented entities can be expressed by a small set of facts (e.g., as in the above example of a university).

6 Concluding Remarks

The families $\mathsf{PrXML}^{\{\mathsf{exp}\}}$ and $\mathsf{PrXML}^{\{\mathsf{cie}\}}$ (of [2, 24]) exhibit a clear tradeoff between the efficiency of query evaluation and the ability to model correlations between probabilistic choices. $\mathsf{PrXML}^{\{\mathsf{exp}\}}$ is the most expressive family among those that do not have cie nodes (see Figure 3). In this family, highly expressive queries can be evaluated efficiently. But this is achieved at the expense of assuming that choices of children by different distributional nodes are independent. In comparison, $\mathsf{PrXML}^{\{\mathsf{cie}\}}$ can express correlations between distributional nodes by means of shared event variables; however, evaluation of queries with projection (even very simple ones) is intractable.

Approximate query evaluation partly surmounts the limitation entailed by the above tradeoff. Specifically, for twig queries with projection, efficient (multiplicative) approximate evaluation is realizable in the most expressive family, namely, $\mathsf{PrXML}^{\{\mathsf{exp},\mathsf{cie}\}}$. But this solution is possible only because twig queries are monotonic. In particular, query evaluation becomes inapproximable if negation can be applied to branches (e.g., "find all departments that do not have a chair"). The PXDB model takes a completely different approach. It describes correlations in a p-document of $\mathsf{PrXML}^{\{\mathsf{exp}\}}$ in terms of a fixed set of constraints (phrased as c-formulae), rather than by many specific dependencies among distributional nodes (as can be done in $\mathsf{PrXML}^{\{\mathsf{cie}\}}$). This is a natural approach, because correlations are quite frequently a facet of integrity constraints. Interestingly, the PXDB approach demonstrates the following phenomenon. When a dependency-free probabilistic data model is coupled with a powerful query language, it becomes a realistic framework that is capable of expressing complex correlations among entities, without sacrificing efficiency.

The above tractability results for $\mathsf{PrXML}^{\{\mathsf{exp}\}}$ hold only under data complexity. Under query-and-data complexity, query evaluation becomes hard even for projection-free twigs over $\mathsf{PrXML}^{\{\mathsf{mux}\}}$. Nevertheless, in [19] it is shown that the maximal matches of a twig pattern in a p-document of $\mathsf{PrXML}^{\{\mathsf{ind},\mathsf{mux}\}}$ w.r.t. a given threshold can be enumerated in incremental polynomial time.

In [2, 24], various aspects of managing probabilistic XML are studied. Their work is couched in the value-based semantics and the focus is on the family $\mathsf{PrXML}^{\{\mathsf{cie}\}}$; in addition, [2] also considers $\mathsf{PrXML}^{\{\mathsf{ind}\}}_{|\mathscr{V}|}$. They model and investigate updates in probabilistic XML. For example, an insertion into a given document is defined by a triple (t, n, d), where t is a twig pattern, n is a node of t and d is a tree that needs

to be added to each node v, such that some match of t in the given document maps n to v. In the setting of probabilistic data, an update modify the possible worlds, and the goal is to represent them by a new p-document. The work of [2, 24] shows how it can be done. Other problems studied in [24] are those of determining whether two p-documents are equivalent, and eliminating random possible worlds characterized by probabilities that are too low. Finally, they consider the problem of applying cardinality constraints to a given p-document and representing the result by means of a new p-document. Their cardinality constraints are a limited, order-unaware form of DTD constraints.

The *PIXml* model of [13] describes probabilistic choices similarly to $\mathsf{PrXML}^{\{\mathsf{exp}\}}_{|\not\perp}$. However, PIXml significantly deviates from p-documents in two aspects. First, the representation of the probability space as well as the possible worlds are directed acyclic graphs, rather than trees. Second, the probabilities of choosing subsets of children are defined by intervals, rather than exact values.

References

[1] S. Abiteboul, B. Kimelfeld, Y. Sagiv, and P. Senellart. On the expressiveness of probabilistic XML models. Submitted for a journal publication, 2008.

[2] S. Abiteboul and P. Senellart. Querying and updating probabilistic information in XML. In *EDBT*, pages 1059–1068, 2006.

[3] S. Amer-Yahia, S. Cho, L. V. S. Lakshmanan, and D. Srivastava. Minimization of tree pattern queries. In *SIGMOD*, pages 497–508, 2001.

[4] O. Benjelloun, A. D. Sarma, A. Y. Halevy, M. Theobald, and J. Widom. Databases with uncertainty and lineage. *VLDB J.*, 17(2):243–264, 2008.

[5] N. Bruno, N. Koudas, and D. Srivastava. Holistic twig joins: optimal XML pattern matching. In *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data*, pages 310–321. ACM, 2002.

[6] S. Cohen, I. Fadida, Y. Kanza, B. Kimelfeld, and Y. Sagiv. Full disjunctions: Polynomial-delay iterators in action. In *VLDB*, pages 739–750. ACM, 2006.

[7] S. Cohen, B. Kimelfeld, and Y. Sagiv. Incorporating constraints in probabilistic XML. In *PODS*, pages 109–118, 2008.

[8] P. Dagum and M. Luby. Approximating probabilistic inference in bayesian belief networks is NP-hard. *Artif. Intell.*, 60(1):141–153, 1993.

[9] N. N. Dalvi and D. Suciu. The dichotomy of conjunctive queries on probabilistic structures. In *PODS*, pages 293–302, 2007.

[10] N. N. Dalvi and D. Suciu. Efficient query evaluation on probabilistic databases. *VLDB J.*, 16(4):523–544, 2007.

[11] R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Monographs in Computer Science. Springer, 1999.

[12] C. A. Galindo-Legaria. Outerjoins as disjunctions. In *SIGMOD*, pages 348–358. ACM Press, 1994.

[13] E. Hung, L. Getoor, and V. S. Subrahmanian. Probabilistic interval XML. In *ICDT*, pages 361–377, 2003.

[14] E. Hung, L. Getoor, and V. S. Subrahmanian. PXML: A probabilistic semistructured data model and algebra. In *ICDE*, pages 467–478, 2003.

[15] M. Jerrum, L. G. Valiant, and V. V. Vazirani. Random generation of combinatorial structures from a uniform distribution. *Theor. Comput. Sci.*, 43:169–188, 1986.

[16] D. S. Johnson, C. H. Papadimitriou, and M. Yannakakis. On generating all maximal independent sets. *Inf. Process. Lett.*, 27(3):119–123, 1988.

[17] R. M. Karp, M. Luby, and N. Madras. Monte-carlo approximation algorithms for enumeration problems. *Journal of Algorithms*, 10(3):429–448, 1989.

[18] B. Kimelfeld, Y. Kosharovsky, and Y. Sagiv. Query efficiency in probabilistic XML models. In *SIGMOD Conference*, pages 701–714, 2008.

[19] B. Kimelfeld and Y. Sagiv. Matching twigs in probabilistic XML. In *VLDB*, pages 27–38, 2007.

[20] B. Kimelfeld and Y. Sagiv. Maximally joining probabilistic data. In *PODS*, pages 303–312. ACM, 2007.

[21] A. Nierman and H. V. Jagadish. ProTDB: Probabilistic data in XML. In *VLDB*, pages 646–657, 2002.

[22] C. H. Papadimitriou and M. Yannakakis. On the complexity of database queries. *Journal of Computer and System Sciences*, 58(3):407–427, 1999.

[23] C. Ré and D.Suciu. Efficient evaluation of HAVING queries on a probabilistic database. In *DBPL*, volume 4797 of *Lecture Notes in Computer Science*, pages 186–200. Springer, 2007.

[24] P. Senellart and S. Abiteboul. On the complexity of managing probabilistic XML data. In *PODS*, pages 283–292, 2007.

[25] S. Toda and M. Ogiwara. Counting classes are at least as hard as the polynomial-time hierarchy. *SIAM J. Comput.*, 21(2):316–328, 1992.

[26] L. G. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8:189–201, 1979.

[27] M. van Keulen, A. de Keijzer, and W. Alink. A probabilistic XML approach to data integration. In *ICDE*, pages 459–470. IEEE Computer Society, 2005.

[28] S. Zachos. Probabilistic quantifiers and games. *Journal of Computer and System Sciences*, 36(3):433–451, 1988.