Updating Probabilistic XML*

Evgeny Kharlamov†
Free University of Bozen-Bolzano
Dominikanerplatz 3
39100 Bozen, Italy
kharlamov@inf.unibz.it

Werner Nutt
Free University of Bozen-Bolzano
Dominikanerplatz 3
39100 Bozen, Italy
nutt@inf.unibz.it

Pierre Senellart
Institut Télécom; Télécom ParisTech
CNRS LTCI, 46 rue Barrault
75634 Paris, France
pierre@senellart.com

ABSTRACT

We investigate the complexity of performing updates on probabilistic XML data for various classes of probabilistic XML documents of different succinctness. We consider two elementary kinds of updates, insertions and deletions, that are defined with the help of a locator query that specifies the nodes where the update is to be performed. For insertions, two semantics are considered, depending on whether a node is to be inserted once or for every match of the query. We first discuss deterministic updates over probabilistic XML, and then extend the algorithms and complexity bounds to probabilistic updates. In addition to a number of intractability results, our main result is an efficient algorithm for insertions defined with branching-free queries over probabilistic models with local dependencies. Finally, we discuss the problem of updating probabilistic XML databases with continuous probability distributions.

Categories and Subject Descriptors

H.2.3 [**Database Management**]: Logical Design, Languages—*data models, query languages*; F.2.0 [**Analysis of Algorithms and Problem Complexity**]: General

General Terms

Algorithms, Theory

Keywords

XML, updates, probabilistic databases, complexity

1. INTRODUCTION

Though traditional database applications, such as bank account management or order processing, have no room for uncertainty, more recent applications, such as information extraction from the World Wide Web [4], automatic schema matching in information integration [17], or information gathering from sensor networks [7] are inherently imprecise. There is a real need for managing in a rigorous way this imprecision. In a number of cases, such as with conditional random fields [15] for information extraction, or uncertain schema mappings [8, 9] in information integration, these automatic imprecise systems provide *probabilities* that the data exists. In other cases, systems do not provide such probabilities, but confidence values, which can sometimes be seen after renormalization as approximate probability values. It makes sense to use these probabilities to represent the confidence the system has in the information, and to manipulate this probabilistic information in a *probabilistic database management system* [5].

Recent work has proposed models for probabilistic data, both in the relational [6, 14, 20] and XML [2, 3, 10, 16] settings. We focus here on the latter, which is particularly adapted in the case, common on the Web, when the information is not strictly constrained by a schema, or when it is inherently tree-like (mailing lists, parse trees of natural language sentences, etc.). A number of works on probabilistic XML have dealt with query answering for a variety of models and query languages [1, 3, 10–13, 16] but updates have received far less attention, with only a handful of works [2, 3, 18] that focus on a specific semantics for updates or on specific models. Updates are, however, an important part of any database management system, and are especially important in the case of probabilistic databases, since probabilistic updates [2] can be seen as the source of probabilities and of the correlations between these probabilities. We propose in this article a general study of the complexity of performing updates in probabilistic XML, for different probabilistic XML models proposed so far. We also briefly discuss the difficulties that arise in updating probabilistic documents with continuous probability distributions of data values [1].

We use a general model for probabilistic XML, from [2, 11], that encompasses various probabilistic XML models from the literature, in the form of *p-documents*, that are defined as trees with ordinary and *distributional* nodes, the latter specifying a probability distribution on the children of a given nodes. Distributional nodes are of different kinds, and we explore the complexity of updates for p-documents that make use of these different types. Following the existing XML update languages, XUpdate [21] and XQuery Update [19], updates (of two kinds: insertions and deletions) are defined

*This research was partially funded by the European Research Council grant Webdam (under FP7), agreement n°226513, and by the Dataring project of the French ANR.

†The author is co-affiliated with INRIA Saclay.

Updates in XML (EDBT Workshop Proceedings), March 22, 2010, Lausanne, Switzerland.

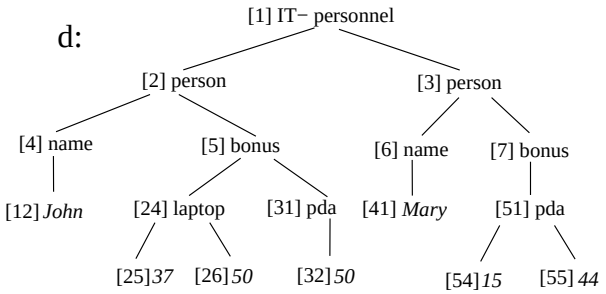

Figure 1: Document d: personnel in IT department

in terms of *locator queries* that specify the nodes where the update is to be performed. The query language obviously has an influence on the complexity of performing these updates, and we distinguish between several different subsets of tree-pattern queries with joins. We also consider two different semantics for insertions: the first one (insert a node at some position *only if* some query has a match) mimics the behavior of XUpdate and XQuery Update, while the second one (insert a node at some position *for all* matches of some query) corresponds to what has been studied in [2].

The contributions of this paper are as follows: (i) a general picture of update tractability for various query languages and probabilistic XML models, with complexity results; (ii) a polynomial algorithm for only-if insertions and deletions defined by descendant-free single-path queries on local models (Theorem 5 and Proposition 8); (iii) the introduction of a very general PrXML^{fie} model, that is especially interesting with respect to updates; (iv) the first discussion of updates in continuous probabilistic XML.

2. DETERMINISTIC DATA AND QUERIES

We assume a countable set of *identifiers* $\mathcal{I}$ and one of *labels* $\mathcal{L}$, such that $\mathcal{I} \cap \mathcal{L} = \varnothing$. A document is a pair $d = (t, \theta)$, where t is a finite, unordered tree over identifiers and θ is a function that maps each node v to a label $\theta(v) \in \mathcal{L}$. Ignoring the ordering of the children of nodes is a common simplification over the XML model that does not significantly change the results of this paper. We use the standard notions *child* and *parent*, *descendant* and *ancestor*, *root* and *leaf* in the usual way. We denote the root of d by $\mathsf{root}(d)$. Two documents d_1 and d_2 are *isomorphic*, denoted by $d_1 \sim d_2$, if there is a one-to-one correspondence between the nodes of the two documents that preserves labels and edges.

Example 1. *Consider the document d from [1], reproduced in Figure 1. Identifiers appear inside square brackets before labels. The document describes the personnel of an IT department and the bonuses distributed for different projects. The document d indicates* `John` *worked under two projects (*`laptop` *and* `pda`*) and got bonuses of* `37` *and* `50` *in the former project and* `50` *in the latter one.*

We introduce tree-pattern queries with joins, with join-free queries and single-path queries as special cases. Let *Var* be a countable set of variables. A *tree pattern* (with joins), denoted Q, is a tree with two types of edges: child-edges, denoted $E_/$, and descendant edges, denoted $E_{//}$. The nodes of the tree are labeled by a labeling function λ with either labels from $\mathcal{L}$ or with variables from *Var*. Variables that occur more than once are called *join variables*. We refer to nodes of Q as n, m in order to distinguish them from the nodes of documents.

A *tree-pattern query with joins* has the form $Q[\bar{n}]$, where Q is a tree pattern with joins and $\bar{n}$ is a tuple of nodes of Q (defining its output). We sometimes identify the query with the pattern and write Q instead of $Q[\bar{n}]$ if $\bar{n}$ is not important or clear from the context. If $\bar{n}$ is the empty tuple, we say that the query is Boolean. A query is *join-free* if every variable in its pattern occurs only once. If the set of edges $E_/ \cup E_{//}$ of a query is a linear order of the nodes, the query is a *single-path query*. We denote the set of all tree pattern queries, which may have joins, as TPJ. The subclasses of join-free and single path queries are denoted as TP and SP, respectively.

A *valuation* γ maps query nodes to document nodes. A document *satisfies* a query if there exists a *satisfying* valuation, or a *match*, mapping query nodes to document nodes in a way that is consistent with edge types, labeling, and variable occurrences. More precisely, (1) nodes connected by child/descendant edges are mapped to nodes that are children/descendants of each other; (2) query nodes with label a are mapped to document nodes with label a; (3) two query nodes with the same variable are mapped to document nodes with the same label. Let q be a TPJ query and d be a document, then $\mathit{Val}(q, d)$ denotes the set of all valuations γ of q in d. Details of query semantics are given in [1].

3. PROBABILISTIC XML

A *finite probability space* over documents, px-*space* for short, is a pair $\mathcal{S} = (D, \mathsf{Pr})$, where D is a finite set of non-isomorphic documents and Pr maps each document to a probability $\mathsf{Pr}(d)$ with $\Sigma\{\mathsf{Pr}(d) \mid d \in D\} = 1$.

p-Documents: Syntax. Following [2], we now introduce a very general syntax for representing compactly px-spaces, called *p-documents*. A p-document is similar to a document, with the difference that it has two types of nodes: *ordinary* and *distributional*. Distributional nodes are only used for defining the probabilistic process that generates random documents (but they do not actually occur in these ones). Ordinary nodes have labels and they may appear in random documents. We require the leaves and the root to be ordinary nodes.

Formally, we assume given a set $\mathcal{X}$ of Boolean random variables with some specified probability distribution Δ over $\mathcal{X}$. A *p-document*, denoted $\widehat{\mathcal{P}}$, is an unranked, unordered, labeled tree. Each node has a unique identifier v and a label $\mu(v)$ in

$$\mathcal{L} \cup \{\mathit{fie}(E)\} \cup \{\mathit{cie}(E)\} \cup \{\mathit{exp}(\mathsf{Pr})\} \cup \{\mathit{mux}(\mathsf{Pr})\} \cup \{\mathit{det}\}$$

where $\mathcal{L}$ are labels of *ordinary* nodes, and the others are labels of *distributional* nodes. We consider different kinds of these distributional labels: $\mathit{fie}(E)$ (for formulas of independent events), $\mathit{cie}(E)$ (for conjunction of independent events), $\mathit{exp}(\mathsf{Pr})$ (for explicit), $\mathit{mux}(\mathsf{Pr})$ (for mutually exclusive), and det (for deterministic). If a node v is labeled with (i) $\mathit{fie}(E)$, then E is a function that assigns to each child of v a propositional formula φ of events from $\mathcal{X}$; (ii) $\mathit{cie}(E)$, then E assigns to each child of v a conjunction $e_1 \wedge \cdots \wedge e_k$ of event literals (x or $\neg x$, for $x \in \mathcal{X}$); (iii) $\mathit{exp}(\mathsf{Pr})$, then Pr assigns to each subset of children of v a probability, summing up to 1; (iv) $\mathit{mux}(\mathsf{Pr})$, then Pr assigns to each child of v a probability,

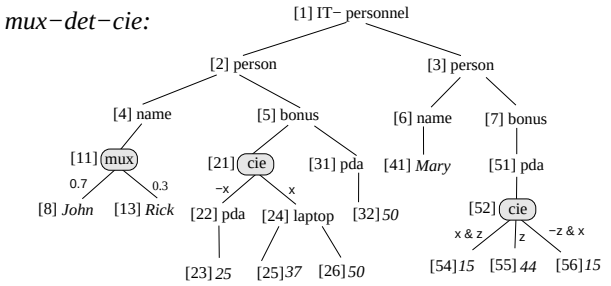

Figure 2: PrXML^{mux,det,cie} p-document: IT dpt.

summing up to 1.

We denote *classes of p-documents* by PrXML with a superscript denoting the types of distributional nodes allowed for the p-documents in the class. For instance, p-documents of PrXML^{mux,det,cie} are with *mux*, *det*, *cie* nodes like in Figure 2.

p-Documents: Semantics. The *semantics of a p-document* $\widehat{\mathcal{P}}$, denoted by $[\![\widehat{\mathcal{P}}]\!]$, is a px-space over *random documents*, where the documents are denoted by $\mathcal{P}$ and are obtainable from $\widehat{\mathcal{P}}$ by a randomized three-step process.

1. We choose a valuation ν of the variables in $\mathcal{X}$. The probability of the choice, according to the distribution Δ, is $p_\nu = \prod_{x \text{ in } \widehat{\mathcal{P}}, \nu(x)=\mathsf{true}} \Delta(x) \cdot \prod_{x \text{ in } \widehat{\mathcal{P}}, \nu(x)=\mathsf{false}} (1-\Delta(x))$.
2. We delete the children v of each *fie*(E) or *cie*(E) node where $\nu(E(v))$ is false, and their descendants. Independently for each *exp*(Pr) (resp., *mux*(Pr)) node v, we select a subset of its children V' (resp., one of its children v') according to the corresponding probability distribution Pr and delete the other children and their descendants, the probability of the choice being $\mathsf{Pr}(V')$ (resp., $\mathsf{Pr}(v')$). We do not delete any of the children of *det* nodes.
3. We then remove in turn each distributional node, connecting each ordinary child v of a deleted distributional node with its lowest ordinary ancestor v', or, if no such v' exists, we turn this child into a root.

The result of this third step is a random document $\mathcal{P}$. The probability $\mathsf{Pr}(\mathcal{P})$ is defined as the product of p_ν, the probability of the variable assignment we chose in the first step, with all $\mathsf{Pr}(V')$ and $\mathsf{Pr}(v')$, the probabilities of the choices that we made in the second step for the *exp* and *mux* nodes.

In addition to the five kinds of distributional nodes presented above (all of which except *fie* from [2]), we occasionally consider *ind*(Pr) nodes, also from [2], which specify independent probabilities for every child of a node. As discussed in [2], *ind* can be simulated with *mux* and *det*.

EXAMPLE 2. *A p-document $\widehat{\mathcal{P}}$ is shown in Figure 2. It has cie, mux and det distributional nodes. For example, node n_{21} has label cie(E) and two children n_{22} and n_{24}, such that $E(n_{22}) = \neg x$ and $E(n_{24}) = x$. Node n_{11} has label mux(Pr) and two children n_8 and n_{13}, such that $\mathsf{Pr}(n_8) = 0.7$ and $\mathsf{Pr}(n_{13}) = 0.3$. Document d in Figure 1 is in $[\![\widehat{\mathcal{P}}]\!]$, and corresponds to the assignment x true, z true, and to the choice of* John *under the mux node. If $\mathsf{Pr}(x) = 1/2$, $\mathsf{Pr}(z) = 1/4$, then $\mathsf{Pr}(d) = 0.7 \cdot 1/2 \cdot 1/4 = 0.0875$.*

Two px-spaces $\mathcal{S}_1 = (D_1, \mathsf{Pr}_1)$ and $\mathcal{S}_2 = (D_2, \mathsf{Pr}_2)$ are *isomorphic*, denoted $\mathcal{S}_1 \sim \mathcal{S}_2$, if they are identical up to isomorphism, i.e., $\sum_{d' \in D_1 : d' \sim d} \mathsf{Pr}_1(d') = \sum_{d' \in D_2 : d' \sim d} \mathsf{Pr}_2(d')$., for all document d. We write $\widehat{\mathcal{P}}_1 \sim \widehat{\mathcal{P}}_2$ if $[\![\widehat{\mathcal{P}}_1]\!] \sim [\![\widehat{\mathcal{P}}_2]\!]$. Two p-documents are *equivalent* if their px-spaces are isomorphic.

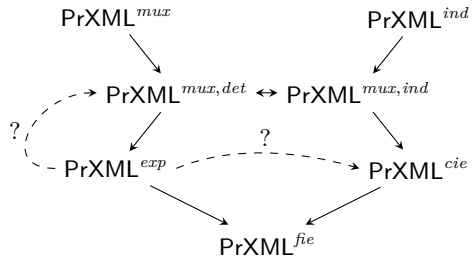

Figure 3: Efficient translations between PrXML families

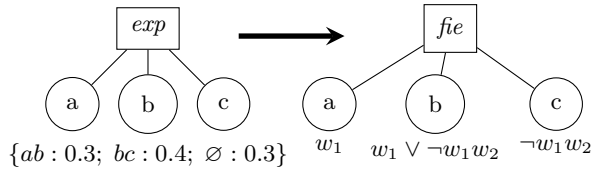

Figure 4: Translation from PrXML^{exp} to PrXML^{fie}, where $p(w_1) = 0.3$; $p(w_2) = 0.4/0.7$

Expressiveness. We are interested in *complete representation systems*: a probabilistic XML model is complete if every px-space can be modeled. It has been shown in [2] that both PrXML^{ind} and PrXML^{mux} are incomplete, whereas PrXML^{mux,det}, PrXML^{exp} and PrXML^{cie} are complete. The PrXML^{fie} model, as a generalization of PrXML^{cie}, is clearly also complete. Our study will focus on these complete representation systems.

Succinctness. It is also important to understand which models can be seen as particular cases of other models or, in other words, if a p-document of a given class can tractably be transformed into an equivalent p-document of another class. This is of help to prove complexity results about the tractability of updates. Figure 3 summarizes the polynomial-time translation between different classes of probabilistic documents. As shown in the picture, it is open whether PrXML^{exp} is polynomially translatable to PrXML^{mux,det} or to PrXML^{cie}. Apart from this, the figure is complete, meaning that the absence of a path between models means there is no efficient translation between them. All the results, except the ones involving PrXML^{fie}, were proved in [2].

Tractable translation from PrXML^{cie} to PrXML^{fie} is clear since the latter is a generalization of the former. The following proposition shows that PrXML^{fie} is exponentially more succinct than PrXML^{exp}.

PROPOSITION 3. *There is a polynomial-time translation from p-documents of* PrXML^{exp} *to equivalent p-documents of* PrXML^{fie}*, but not the other way around.*

PROOF. (Sketch) A translation from PrXML^{exp} to PrXML^{fie} is illustrated on an example document in Figure 4, and can be generalized in a straightforward way. The other direction is clear since there is no efficient translation from PrXML^{cie}

to PrXML^{exp} [2]. □

It is actually possible (but slightly more involved) to show PrXML^{fie} is exponentially more succinct than $\mathsf{PrXML}^{exp,cie}$, which is a stronger result.

We next investigate updates for the four complete PrXML models presented in Figure 3 that are different in succinctness: $\mathsf{PrXML}^{mux,det}$, PrXML^{exp}, PrXML^{cie} and PrXML^{fie}.

4. DETERMINISTIC UPDATES

A *deterministic update*, or simply *update*, is a triple (q, n, t) also denoted $q^{n,t}$, where (i) q is a TPJ query, called the *condition* of the update; (ii) n is a node of q, called the *locator* of the update; (iii) t is a document such that $nodes(t) \cap nodes(q) = \varnothing$ and possibly with variables $Var(t)$ among its labels, with $Var(t) \subseteq Var(q)$. An update is *variable-free* if t is a document, i.e., $Var(t) = \varnothing$.

The intuition behind the semantics of updates is the following: if q is satisfied in a document d, then one either inserts t in d as a child of a node corresponding to n, or deletes all nodes corresponding to n together with all their descendants. The way how to insert t depends on the specific semantics of insertions and will be discussed below. For deletions, t is irrelevant and we just denote q^n.

Depending on the type of the query q we distinguish the following classes of updates: TPJ, TP, SP and RSP, where RSP is the subclass of SP where n is the (unique) leaf of q.

Only-if insertions. This semantics addresses updates that are insertions of the kind: "For every professor, insert a bonus of 5 *only if* her team is in some EU project."

Formally, let $q^{n,t}$ be a variable-free update and d a document (only-if semantics is only defined for insertions with variable-free t). A *deterministic only-if insertion* $oi(q^{n,t}, d)$ in d by $q^{n,t}$ is a document d' obtained from d by rooting t to every $v \in d$ such that $v = \gamma(n)$ for some $\gamma \in Val(q, d)$. Note that t can be rooted to each node of d at most once. Moreover, if there is no valuation of q in d, then $oi(q^{n,t}, d) = d$.

For-all insertions. This semantics addresses updates that are insertions of the kind: "For every professor, insert a bonus of X *for all* EU projects with a duration of X years, that her team is involved in." This is the semantics that was considered in [2], except that only variable-free documents were allowed.

Formally, let $q^{n,t}$ be an update and d be a document. A *deterministic for-all insertion* $fa(q^{n,t}, d)$ in d by $q^{n,t}$ is a document d' obtained from d by rooting to each $v \in d$, such that $v = \gamma(n)$ for some $\gamma \in Val(q, d)$, the forest

$$\{\!\{\gamma'(t) \mid \gamma' \in Val(q, d) \text{ and } \gamma'(n) = v\}\!\}.$$

In other words, instantiated versions of t can be rooted to each node of d multiple times, as many times as the number of valuations of the query in d. Again, if there is no valuation of q in d, then $fa(q^{n,t}, d) = d$.

Deterministic insertions in p-documents. We now extend the definition of deterministic updates to px-spaces. Let $q^{n,t}$ be an update and $\mathcal{S}$ be a px-space. Then a *deterministic for-all insertion* $fa(q^{n,t}, \mathcal{S})$ in $\mathcal{S} = \{(d_i, \mathsf{Pr}(d_i)) \mid 1 \leqslant i \leqslant n\}$ by $q^{n,t}$ is the px-space:

$$\{(d', \mathsf{Pr}'(d')) \mid d' = fa(q^{n,t}, d), \text{ for some } (d, \mathsf{Pr}(d)) \in \mathcal{S}, \text{ and } \mathsf{Pr}'(d') = \sum_{\substack{d \in \mathcal{S} \\ fa(q^{n,t}, d) \sim d'}} \mathsf{Pr}(d)\}. \quad (1)$$

We define the only-if insertions $oi(q^{n,t}, \mathcal{S})$ for updates $q^{n,t}$ with variable-free t analogously to $fa(q^{n,t}, \mathcal{S})$. Finally, we define insertions for p-documents as $fa(q^{n,t}, \widehat{\mathcal{P}}) = fa(q^{n,t}, [\![\widehat{\mathcal{P}}]\!])$ and $oi(q^{n,t}, \widehat{\mathcal{P}}) = oi(q^{n,t}, [\![\widehat{\mathcal{P}}]\!])$.

Deterministic deletions in p-documents. Let q^n be an update and $\mathcal{S}$ be a px-space. Then a *deterministic deletion* $del(q^n, d)$ *in a document* d is the document obtained from d by deleting all nodes $\gamma(n)$ and their descendants, where γ in $Val(q, d)$.

Deterministic deletions for px-spaces are defined as in (1), by substituting "fa" with "del" and "$q^{n,t}$" with "q^n". For p-documents $\widehat{\mathcal{P}}$, we define $del(q^n, \widehat{\mathcal{P}}) = del(q^n, [\![\widehat{\mathcal{P}}]\!])$.

Problems to Investigate. Let $\mathcal{D}$ be a family of p-documents and $\mathcal{Q}$ a class of TPJ queries. We only consider *data complexity*, i.e., the query is not considered to be part of the input. We say that $\mathcal{D}$ is *closed under*, say, deterministic fa-insertions for $\mathcal{Q}$ if, for any deterministic update $q^{n,t}$ where $q \in \mathcal{Q}$, and for each $\widehat{\mathcal{P}} \in \mathcal{D}$ there exists a $\widehat{\mathcal{P}}' \in \mathcal{D}$ such that $\widehat{\mathcal{P}}' \sim fa(q^{n,t}, \widehat{\mathcal{P}})$. The closure is *tractable* if $\widehat{\mathcal{P}}'$ can be computed in time polynomial in the size of $\widehat{\mathcal{P}}$. The closure is *linear* if $|\widehat{\mathcal{P}}'| = O(|\widehat{\mathcal{P}}| \cdot |t|)$ for all $q \in \mathcal{Q}$, that is, there is a constant $C > 0$ such that $|fa(q^{n,t}, \widehat{\mathcal{P}})| \leqslant C \cdot |\widehat{\mathcal{P}}| \cdot |t|$ for all $q^n \in \mathcal{Q}$, $\widehat{\mathcal{P}} \in \mathcal{D}$, and documents t. The closure is *not in* PSPACE if the computation of p-documents $\widehat{\mathcal{P}}'$ (in the worst case) requires more than polynomial space in the size of $\widehat{\mathcal{P}}$. Finally, the closure is #P-*hard* if there is a polynomial-time (Turing) reduction from a #P-hard problem to the problem of computing a p-document $\widehat{\mathcal{P}}'$ such that $[\![\widehat{\mathcal{P}}']\!] \sim fa(q^{n,t}, \widehat{\mathcal{P}})$. Recall that #P is the class of functions that count the number of accepting paths of an NP Turing machine, which is conventionally deemed intractable. We analogously define the same notions for oi-insertions and deletions.

If a closure is linear for a class of queries and p-documents, then a sequence of updates can lead at most to exponential growth of the original document. More precisely, if the updates $q_i^{n_i,t_i}$, with $1 \leqslant i \leqslant n$ are applied one after the other to a p-document $\widehat{\mathcal{P}}$, resulting in some $\widehat{\mathcal{P}}'$, then $|\widehat{\mathcal{P}}'| \leqslant C^n |\widehat{\mathcal{P}}| \cdot |t_1| \cdots |t_n|$.

Clearly, exponential growth of documents is not desirable. However, in principle tractability of closure does not exclude the possibility of still larger growth. To see this, note that a quadratic increase of the document size after each update results in document growth that is doubly exponential.

5. TRACTABLE CLOSURE WRT UPDATES

We explore in this section tractability results for deterministic updates. We now introduce a technical term needed further in the proofs. One can extend the notion of matches of queries $q \in$ TPJ over documents to p-documents by considering mappings that ignore distributional nodes and are still consistent with the edge types, the labeling, and the variable occurrences in q. We call these matches *naive*. If γ

is a naive match of q in $\widehat{\mathcal{P}} \in \mathsf{PrXML}^{fie}$, then the *formula set* of the math γ over q is the set of all the formulas that occur on the edges of the sub p-document $\gamma(q)$ of $\widehat{\mathcal{P}}$ "covered" by q after applying γ.

Let us start with a straightforward result about the link between the *only-if* and *for-all* semantics of insertions when they are used on a very limited query language.

LEMMA 4. *For insertions with descendant-free* RSP *conditions, only-if and for-all semantics coincide for p-documents of any considered type.*

Only-if insertions	PrXML Model			
	mux,det	*exp*	*cie*	*fie*
RSP	L	L	L	L
SP	P*	P*	#P-hard	L
TP	?	?	#P-hard	P
TPJ	#P-hard	#P-hard	#P-hard	P

Table 1: Tractable closure under deterministic only-if insertions. L stands for linear and P for polynomial. * for descendant-free SP only.

Only-if insertions. We now investigate the complexity of only-if insertions. The following result is somewhat involved and shows that models with local dependencies behave nicely with respect to simple only-if insertions.

THEOREM 5. *For* RSP *insertions under only-if semantics, both* PrXML^{exp} *and* $\mathsf{PrXML}^{mux,det}$ *are linearly closed. If conditions are in* SP *and descendant-free, both classes are tractably closed.*

PROOF. If $q^{n,t}$ is an RSP insertion and there is a naive match γ of q in $\widehat{\mathcal{P}} \in \mathsf{PrXML}^{mux,det} \cup \mathsf{PrXML}^{exp}$, then one inserts t under $\gamma(n)$ and iterates the insertion for all such γ that are distinct in n. The resulting p-document is the update $oi(q^{n,t}, \widehat{\mathcal{P}})$ and it has linear size wrt $\widehat{\mathcal{P}}$ since there are only linearly many naive matches of q in $\widehat{\mathcal{P}}$.

Let $q^{n,t}$ be a descendant-free SP insertion and $\widehat{\mathcal{P}}$ be in $\mathsf{PrXML}^{mux,det}$ (the generalization to PrXML^{exp} is straightforward). Without loss of generality, assume $\widehat{\mathcal{P}}$ has at most two children for every *mux* node (any $\widehat{\mathcal{P}}$ can be transformed in such a form in polynomial time). Let V be the set of all nodes v of $\widehat{\mathcal{P}}$ such that there is a naive matches γ with $\gamma(n) = v$. Intuitively, V contains the nodes where the insertion can take place.

For every $v \in V$, the insertion $q^{n,t}$ should add t as a child of v, but only in some of the possible worlds of $\widehat{\mathcal{P}}$. For $v \in V$, let $\widehat{\mathcal{P}}_v$ be a p-subdocument of $\widehat{\mathcal{P}}$ rooted at v, and $\gamma_1, \ldots, \gamma_m$ the set of all naive matches of q in $\widehat{\mathcal{P}}$ such that $\gamma_i(n) = v$ for every $1 \leqslant i \leqslant m$. Then the worlds d where $q^{n,t}$ should bring t under v are exactly those where (at least) one of the γ_i's matches d.

In order to describe the resulting px-space with a p-document $\widehat{\mathcal{P}}'$, we process v by (i) inserting a *mux* node v' under it, and then (ii) rooting n p-documents $\widehat{\mathcal{P}}_1^v, \cdots, \widehat{\mathcal{P}}_m^v$ under v', one per match γ_i. Every $\widehat{\mathcal{P}}_i^v$ is a modification of $\widehat{\mathcal{P}}_v$ where (a) γ_i is *certain* and (b) the $\widehat{\mathcal{P}}_i^v$'s define disjoint px-spaces. We describe further this construction below.

(a) Let γ_i^v be the restriction of γ_i to the nodes of $\widehat{\mathcal{P}}_v$, and $v_1, \cdots, v_{k_i}$ be the nodes of $\widehat{\mathcal{P}}_v$, that are in the image of $\gamma_i^v(q)$ and all have *mux* parents. Let v'_j be the parent of v_j for $j = 1, \ldots, k_i$. Then we replace each of these *mux* nodes v'_j by v_j and drop the (possible) second child of v'_j. The resulting p-document $\widehat{\mathcal{P}}_v'^i$ represents documents that contain $\gamma_i^v(q)$, that is, γ_i^v is *certain* in it. We set the probability $p_i'^v$ of $\widehat{\mathcal{P}}_v'^i$ as the product of the probabilities of all choices performed.

(b) Now we modify every $\widehat{\mathcal{P}}_v'^i$ in order to make them disjoint. We do this by *obstructing* in every $\widehat{\mathcal{P}}_v'^i$ all mappings $\gamma_1^v, \cdots, \gamma_{i-1}^v$; the resulting, *obstructed*, p-documents is denoted by $\widehat{\mathcal{P}}_v^i$. The obstruction works as follows. To obstruct a mapping γ_j^v in $\widehat{\mathcal{P}}_v'^i$, for some $j \leqslant i$, we look for the *mux* node v'' in $\widehat{\mathcal{P}}_i'^v$ covered by γ_j^v that is the most remote from $\mathsf{root}(\widehat{\mathcal{P}}_i'^v)$, and we choose the child of v'' that is not covered by γ_j^v with the probability q_j defined by v''. Then we connect the chosen node to its lowest surviving ancestor and delete from $\widehat{\mathcal{P}}_i'^v$ the *mux* node v'' and its child that was not chosen together with all its descendants. Obstructing in $\widehat{\mathcal{P}}_v'^i$ all the mappings $\gamma_1^v, \ldots, \gamma_{i-1}^v$ results in $\widehat{\mathcal{P}}_v^i$ and its probability is $p_i^v = p_i'^v \cdot \prod_{j=1}^{i-1} p_j$. It is easy to see that the $\widehat{\mathcal{P}}_v^i$'s define disjoint px-spaces and $\widehat{\mathcal{P}}_v^1 = \widehat{\mathcal{P}}_v'^1$. In some cases it might be impossible to obstruct a mapping γ_j in $\widehat{\mathcal{P}}_i'^v$. This means that we cannot make $\widehat{\mathcal{P}}_i'^v$ to be disjoint from $\widehat{\mathcal{P}}_j'^v$, which happens if $[\![\widehat{\mathcal{P}}_i'^v]\!] \subseteq [\![\widehat{\mathcal{P}}_j'^v]\!]$. Let $\widehat{\mathcal{P}}_{i_1}^v, \cdots, \widehat{\mathcal{P}}_{i_k}^v$ be all the obstructions from $\widehat{\mathcal{P}}_1'^v, \ldots, \widehat{\mathcal{P}}_m^v$. Then we root them under v' and the probability of the edge going to $\widehat{\mathcal{P}}_{i_j}^v$ is $p_{i_j}^v$, with the resulting document denoted as $\widehat{\mathcal{P}}'$.

One should also take into account the case when there is no valuation of q in some of $[\![\widehat{\mathcal{P}}]\!]$, that is, when the probability of the query is not 1. To do so, one obstructs in $\widehat{\mathcal{P}}$ all naive matches of q in $\widehat{\mathcal{P}}$ (let the probability of the obstruction be p), and roots the resulting p-document together with $\widehat{\mathcal{P}}'$ under a common *mux* node, with the probability p on the edge to the former p-document and $1 - p$ to the latter.

We thus obtain a p-document that is by construction isomorphic to $oi(q^{n,t}, \widehat{\mathcal{P}})$. The fact the query is SP is critical for the construction of the obstructions to be polynomial. □

We complete the picture for only-if insertions with the following result, obtaining this way the summary of Table 1. Note that the case of TP queries and SP queries with descendant edges remains open for $\mathsf{PrXML}^{mux,det}$ and PrXML^{exp}.

PROPOSITION 6. *For the only-if semantics of insertions,*

1. *the class* PrXML^{cie} *is tractably closed under* RSP *and the closure is linear;*
2. SP *insertions are* #P*-hard for* PrXML^{cie}*, while* TPJ *insertions are* #P*-hard for* $\mathsf{PrXML}^{mux,det}$ *and* PrXML^{exp}*;*
3. *the class* PrXML^{fie} *is tractably closed under* RSP*,* SP*,* TP *and* TPJ *and the closure is linear for* RSP *and* SP*.*

PROOF.

1. Analogous to the case of RSP updates in Theorem 5.

2. By reduction from the computation of the probability that a Boolean SP query matches a PrXML^{cie} document, which is #P-hard [12], as is that of a Boolean TPJ query matching a $\mathsf{PrXML}^{mux,det}$ document [1].

Let $q^{n,t}$ be an update, where q is a SP query, n is the root of q and t is a single-node document $v \notin nodes(q)$. Then for

For-all insertions	PrXML Model			
	mux,det	*exp*	*cie*	*fie*
RSP	L/P	L/P	L/P	L/P
SP	not in PSPACE [2]		L/P	L/P
TP	not in PSPACE		P	P
TPJ	not in PSPACE, #P-hard		P [2]	P

Table 2: Tractable closure under deterministic for-all insertions. L stands for linear and P for polynomial. L/P means linear for descendant-free queries, polynomial otherwise.

Deletions	PrXML Model			
	mux,det	*exp*	*cie*	*fie*
RSP	L	L	L	L
SP	P*	P*	#P-hard	L
TP	?	?	#P-hard	P
TPJ	#P-hard	#P-hard	#P-hard	P

Table 3: Tractable closure under deterministic deletions. L stands for linear and P for polynomial. * for descendant-free SP only.

any *cie* document $\widehat{\mathcal{P}}$, the p-document $\widehat{\mathcal{P}}' = oi(q^{n,v}, \widehat{\mathcal{P}})$ that represents the update should have some (possible none) *cie* nodes under its root, to which v is rooted (since $n = \mathsf{root}(q)$) and the conjunctions of events that label edges going to v are all disjoint (since any $d \in [\![\widehat{\mathcal{P}}']\!]$ has at most one v connected to its root). Due to this disjointness and the fact that we can compute in polynomial time the probability that a given *node* occurs in a world of a *cie* p-document (just compute the probability of the conjunctions of events on the way to the root), we can also compute in polynomial time the probability p that the *label* v occurs in a world of $\widehat{\mathcal{P}}'$. Since p is the same as the probability that q matches $\widehat{\mathcal{P}}$, its computation is a #P-complete problem. Hence, polynomial construction of $\widehat{\mathcal{P}}'$ implies #P-hardness of the update.

The case of TPJ queries and $\mathsf{PrXML}^{mux,det}$ (or PrXML^{exp}) is even simpler, since the probability of TP queries can be computed in polynomial time in such models [12].

3. Let $q^{n,t}$ be an SP update. If there is a naive match γ of q in $\widehat{\mathcal{P}} \in \mathsf{PrXML}^{fie}$, then let Γ be the set of all γ' such that $\gamma'(n) = \gamma(n)$. If φ_γ is the conjunction of all the conditions of the match $\gamma(q)$, then one inserts a *fie* node v under $\gamma(n)$ and roots t under v. The formula that labels the edge from v to t is $\vee_{\gamma\in\Gamma}\varphi_\gamma$. One iterates the insertion for all such γ that are distinct in n. The number of nodes inserted in $\widehat{\mathcal{P}}$ is at most $(|t|+1)\cdot|nodes(\widehat{\mathcal{P}})|$, and the number of formulas occurring under introduced *cie* nodes is at most $|formulas(\widehat{\mathcal{P}})|$. Hence, the construction of $oi(q^{n,t}, \widehat{\mathcal{P}})$ is linear.

For the classes TP and TPJ the proof is similar to the case of SP and the case of RSP is subsumed by SP. For TP and TPJ the updated p-document $oi(q^{n,t}, \widehat{\mathcal{P}})$ is not linear due to the fact that in some cases the number of formulas occurring under introduced *cie* nodes can be quadratic in $|formulas(\widehat{\mathcal{P}})|$. □

For-all insertions. We now consider the tractability of for-all insertions, for which some of the results come from [2], and some are novel, as shown on Table 2.

Proposition 7. *For the for-all semantics of insertions,*

1. *the classes* $\mathsf{PrXML}^{mux,det}$ *and* PrXML^{exp} *are tractably closed under* RSP*, moreover the closure is linear for descendant-free locator queries;*
2. TPJ *insertions are* #P*-hard for the classes* $\mathsf{PrXML}^{mux,det}$ *and* PrXML^{exp}*;*
3. *the classes* PrXML^{cie} *and* PrXML^{fie} *are tractably closed under* RSP *and* SP*,* TP *and* TPJ*, and the closure is linear for* RSP *and* SP *with descendant-free locator queries.*

Proof.

1. Straightforward. The linear bound holds due to Proposition 4 and Theorem 5.

2. Let $\widehat{\mathcal{P}}$ be a p-document and q be a TPJ query. Let a be a label that does not occur in either $\widehat{\mathcal{P}}$ or q. Let $q^{n,t}$ be an update such that n is the root of q and t is a tree with a single node labeled a. If we can compute the insertion $\widehat{\mathcal{P}}'' \sim fa(q^{n,t}, \widehat{\mathcal{P}})$ in P, then we can also compute the probability that a is in $\mathcal{P}'$ in P [11]. But this probability is the probability that the TPJ query q matches $\widehat{\mathcal{P}}$. Computation of the probability that a TPJ query matches a p-document is #P-hard [1], hence the statement is proved.

3. For PrXML^{cie} and SP the proof can be based on the same techniques as in Case 3 of Proposition 6. In short, one inserts a new *cie* node v under $\gamma(n)$ for every distinct $\gamma' \in \Gamma$ and roots $\gamma'(t)$ under v. The formula marking the edges from v to $\mathsf{root}(\gamma'(t))$ is $\varphi_{\gamma'}$. Again linearity follows from the fact that there are linearly many naive matches of SP descendant-free queries in p-documents.

For PrXML^{cie} and TP, TPJ the proof is analogous to the one for SP. The linearity is lost because the number of valuations is polynomial and the number of introduced events is quadratic.

For PrXML^{fie} the proof is the same as for PrXML^{cie}. □

Deletions. We now study the complexity of deleting nodes defined by a locator query. The results of the following proposition are summarized in Table 3 and are exactly the same as for only-if insertions. The tractability of deletions in local models is still open for SP with descendant edges and TP queries.

Proposition 8.

1. $\mathsf{PrXML}^{mux,det}$*,* PrXML^{exp}*,* PrXML^{cie}*, and* PrXML^{fie} *are tractably closed under* RSP *deletions and the closure is linear.*
2. $\mathsf{PrXML}^{mux,det}$ *and* PrXML^{exp} *are tractably closed under descendant-free* SP *deletions.*
3. TPJ *deletions are* #P*-hard for the classes* $\mathsf{PrXML}^{mux,det}$*,* PrXML^{exp}*.*
4. SP *deletions are* #P*-hard for* PrXML^{cie}*.*
5. PrXML^{fie} *is tractably closed under* SP*,* TP *and* TPJ *deletions and* SP *deletions are linear.*

Proof.

1, 2. The proof is analogous to the one for the first part of Theorem 5.

3. Let $\widehat{\mathcal{P}}$ be a p-document in $\mathsf{PrXML}^{mux,det}$ and q be a TPJ query. Let a and b labels that occur neither in $\widehat{\mathcal{P}}$ nor in q.

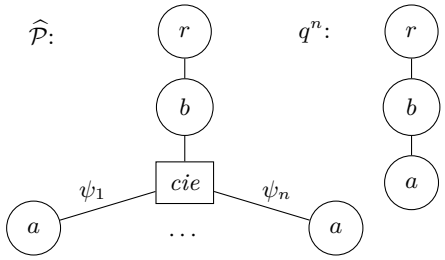

Figure 5: #P-hardness of SP deletions for PrXML^{cie}.

Then one can construct in constant time (i) a p-document $\widehat{\mathcal{P}}'$ from $\widehat{\mathcal{P}}$ by adding a child labeled a to the root of $\widehat{\mathcal{P}}$ and a child labeled b to a, and (ii) an update q'^n by adding a child labeled a to the root of q and a child n labeled b to a. If we can compute the deletion $\widehat{\mathcal{P}}'' = del(q'^n, \widehat{\mathcal{P}}')$ in polynomial time, then we can also compute the probability p that a is a leaf of $\mathcal{P}'$ in polynomial time due to the results of [11]. At the same time p is the probability that the TPJ query q matches $\widehat{\mathcal{P}}$, which is a #P-hard problem.

4. #P-hardness can be shown by reduction from #DNF, using the p-document and update in Figure 5, where n is labeled with b.

5. Let q^n be an update and $\widehat{\mathcal{P}}$ be a p-document in PrXML^{fie}. For a naive match γ of q in $\widehat{\mathcal{P}}$, one constructs Γ as in the proof of Case 3 in Proposition 6. Assume that v is the parent of $\gamma(n)$ in $\widehat{\mathcal{P}}$. Then one modifies $\widehat{\mathcal{P}}$ as follows: one deletes the edge between v and $\gamma(n)$ in $\widehat{\mathcal{P}}$ and introduces a new *fie* node as a child of v and the parent of $\gamma(n)$. The formula that labels the edge to $\gamma(n)$ is $\neg \bigvee_{\gamma' \in \Gamma} \varphi_{\gamma'}$, where $\varphi_{\gamma'}$ is the formulas in the formula set of γ'. If one iterates the modification over all γ that are different in n, one obtains a p-document $\widehat{\mathcal{P}}'$. It is easy to see that $\widehat{\mathcal{P}}' = del(q^n, \widehat{\mathcal{P}})$.

If q is in RSP or SP then $\widehat{\mathcal{P}}'$ has size linear in $\widehat{\mathcal{P}}$, since there are only linearly many naive matches γ and the overall size of all the formulas introduced into $\widehat{\mathcal{P}}'$ is also linear in $|events(\widehat{\mathcal{P}})|$. This proves that the closure is linear for RSP and SP.

If q is in TP or TPJ then the overall size of all the formulas introduced into $\widehat{\mathcal{P}}'$ is in the worst case quadratic in $|events(\widehat{\mathcal{P}})|$. This proves that the closure is tractable for TP and TPJ. □

6. PROBABILISTIC UPDATES

We now discuss a probabilistic variant of updates. An example is, for example, "with a confidence 1/3, make a bulk insertion of a bonus equal to 5 for every professor *only if* her team is in some EU project." Probabilistic updates are also important because they can be seen as a way to obtain probabilistic documents from regular documents.

A *probabilistic update* is a pair $(q^{n,t}, p)$, which we also denote $q^{n,t,p}$, where $q^{n,t}$ is an update and $p \in (0, 1]$ is a rational number, called the *confidence* in the update. Intuitively, the confidence defines the probability the update operation is carried out.

Let $q^{n,t,p}$ be a variable-free probabilistic update and d be a document. The *probabilistic only-if insertion* $oi(q^{n,t,p}, d)$ in d by $q^{n,t,p}$ is the px-*space* $\{(d', p), (d, 1-p)\}$, where $d' = oi(q^{n,t}, d)$. If there is no valuation of q in d, then $oi(q^{n,t,p}, d) = \{(d, 1)\}$.

The semantics of probabilistic *fa*-insertions is defined similarly to probabilistic *oi*-insertions, that is, $fa(q^{n,t,p}, d) := \{(d', p), (d, 1-p)\}$, with the difference that $d' = fa(q^{n,t}, d)$.

We now extend the definition of probabilistic updates to px-spaces. Let $q^{n,t,p}$ be a probabilistic update and $\mathcal{S}$ be a px-space. Then the *probabilistic for-all insertion* $fa(q^{n,t,p}, \mathcal{S})$ in $\mathcal{S} = \{(d_i, \mathsf{Pr}(d_i)) \mid i = 1, \ldots, n\}$ by $q^{n,t,p}$ is the px-space:

$$\{(d', \mathsf{Pr}'(d')) \mid d' \in \mathcal{S} \text{ or } d' = fa(q^{n,t}, d), \text{ for } (d, \mathsf{Pr}(d)) \in \mathcal{S}, \text{ and } \mathsf{Pr}'(d') = p \times \sum_{\substack{d \in \mathcal{S} \\ fa(q^{n,t}, d) \sim d'}} \mathsf{Pr}(d) + (1-p) \times \sum_{\substack{d \in \mathcal{S} \\ d \sim d'}} \mathsf{Pr}(d)\}. \quad (2)$$

The only-if insertions $oi(q^{n,t,p}, \mathcal{S})$ are defined for updates $q^{n,t,p}$ with variable-free q and analogous to $fa(q^{n,t,p}, \mathcal{S})$ with the difference that *fa* should occur in (2) instead of *oi*. Finally, we define updates for p-documents $fa(q^{n,t,p}, \widehat{\mathcal{P}}) = fa(q^{n,t,p}, [\![\widehat{\mathcal{P}}]\!])$ and $oi(q^{n,t,p}, \widehat{\mathcal{P}}) = oi(q^{n,t,p}, [\![\widehat{\mathcal{P}}]\!])$.

The syntax and semantics of probabilistic deletions are defined similarly. Note that there is a crucial difference between deterministic and probabilistic updates of a px-space $\mathcal{S}$. The former *never increase* the cardinality of the resulting space $\mathcal{S}'$, that is $|\mathcal{S}'| \leqslant |\mathcal{S}|$, because $\mathcal{S}'$ consists of the updated versions d' of all documents $d \in \mathcal{S}$. The latter *never decrease* the cardinality of the resulting space $\mathcal{S}'$, that is $|\mathcal{S}| \leqslant |\mathcal{S}'|$, because the resulting space consists of both: all the documents of $\mathcal{S}$ and the updated documents d' for every document $d \in \mathcal{S}$.

We extend the definitions of closure and tractable closure from deterministic to probabilistic updates in a natural way. In terms of tractable closure, probabilistic updates affects all classes of p-documents in the same way as deterministic updates.

PROPOSITION 9. *Let $\mathcal{Q}$ be one of* RSP, SP, TP, *or* TPJ, *and $\mathcal{D}$ one of* PrXML^{mux,det}, PrXML^{exp}, PrXML^{cie} *or* PrXML^{fie}.

1. *If for all conditions from $\mathcal{Q}$, the class $\mathcal{D}$ is tractably closed under deterministic insertions or deletions, for only-if or for-all semantics, then the same is true for D under probabilistic insertions or deletions, respectively, for that semantics.*
2. *For $\mathcal{D}$, hardness of closure under $\mathcal{Q}$ only-if (for-all) deterministic insertions or deletions implies hardness for probabilistic $\mathcal{Q}$ only-if (for-all) insertions or deletions, respectively.*

PROOF. (Sketch) We give an intuition of the proof for probabilistic *oi*-insertions. For other updates the proof is analogous. If we can compute $\widehat{\mathcal{P}}' = oi(q^{n,t}, \widehat{\mathcal{P}})$ in linear time, then we construct a p-document $\widehat{\mathcal{P}}''$ that "gathers" both $\widehat{\mathcal{P}}$ and $\widehat{\mathcal{P}}'$ under a common *mux* root (or the analogue in the other p-document families) with an edge to $\widehat{\mathcal{P}}'$ labeled by p, and to $\widehat{\mathcal{P}}$ labeled by $1-p$. The constructed $\widehat{\mathcal{P}}''$ is exactly $oi(q^{n,t,d}, \widehat{\mathcal{P}})$ and is of linear size in $|\widehat{\mathcal{P}}|$. □

7. UPDATING CONTINUOUS PROB. XML

Many applications of probabilistic databases require the possibility of representing continuous probability distributions. For example, imprecision on the measurement of a sensor may be modeled by a normal distribution centered at

the measured value, and a totally unknown value between 0 and 1 by a uniform distribution. We have introduced in [1] the semantics for another kind of distributional node, *cont*, that can be used to specify that a leaf of the tree follows a continuous distribution of a given type (Gaussian, uniform, Poisson, etc.), independently of the distribution of other leaves. In this section, we briefly discuss the complications that arise when updating continuous p-documents.

Consider the very simple p-document $\widehat{\mathcal{P}} \in \mathsf{PrXML}^{cont}$ that consists of a root with two ordinary children, labeled a and b, each of them having an identical continuous child having uniform distribution between 0 and 1. First, note that the query languages that were defined in Section 2 are not really meaningful in the context of continuous probabilistic documents: the probability of any query imposing a condition on the value of the continuous leaves, or the probability of a value-based join, is 0 since the probability to pick any given constant at random is typically 0. More interesting query languages for continuous data involve range queries. Consider for instance the Boolean (tree-pattern-with-join) range query Q: "Is there an a node whose child has value greater than that of a b node?", and the only-if insertion i that adds a node labeled by c as child of the root if Q matches. The semantics of Q is quite clear, and one can compute the probability that Q matches $\widehat{\mathcal{P}}$, which is $\frac{1}{2}$. On the other hand, it is possible to show that there is no p-document $\widehat{\mathcal{P}}'$ of $\mathsf{PrXML}^{cont,cie}$ that represents the result of applying i to $\widehat{\mathcal{P}}$. Intuitively, this is because in the result document the continuous distributions under a and b nodes are correlated. It is therefore necessary to have a syntax and semantics for correlated continuous distributions (perhaps as an algebra over independent continuous variables) to represent the result of an update over continuous probabilistic documents.

8. CONCLUSION

We have discussed the tractability of updates in a variety of probabilistic XML models. The situation is a bit more complex than the one presented in [2]: if it is true that models with arbitrary dependencies (*cie*, *fie*) can usually express the result of an update more concisely than local models, at the cost of higher complexity of queries, there are cases (only-if descendant-free SP insertions and deletions) where it is possible to efficiently apply an update to local models and not to a *cie* document. There are a couple of open issues with respect to the tractability of only-if insertions and deletions. More importantly, an interesting question is the possibility of applying a sequence of updates in a more efficient way than with the exponential algorithm that is given by the simple iteration of the algorithms for elementary updates. Finally, the results of this paper can be extended to insertions of probabilistic documents, either given as constant or obtained as part of the query match.

References

[1] S. Abiteboul, T.-H. H. Chan, E. Kharlamov, W. Nutt, and P. Senellart. Aggregate queries for discrete and continuous probabilistic XML. In *Proc. ICDT*, Lausanne, Switzerland, Mar. 2010.

[2] S. Abiteboul, B. Kimelfeld, Y. Sagiv, and P. Senellart. On the expressiveness of probabilistic XML models. *VLDB Journal*, 18(5):1041–1064, Oct. 2009.

[3] S. Abiteboul and P. Senellart. Querying and updating probabilistic information in XML. In *Proc. EDBT*, Munich, Germany, Mar. 2006.

[4] C.-H. Chang, M. Kayed, M. R. Girgis, and K. F. Shaalan. A survey of Web information extraction systems. *IEEE Transactions on Knowledge and Data Engineering*, 18(10):1411–1428, Oct. 2006.

[5] N. N. Dalvi, C. Ré, and D. Suciu. Probabilistic databases: Diamonds in the dirt. *Communications of the ACM*, 52(7), 2009.

[6] N. N. Dalvi and D. Suciu. Management of probabilistic data: foundations and challenges. In *Proc. PODS*, Beijing, China, June 2007.

[7] A. Deshpande, C. Guestrin, S. Madden, J. M. Hellerstein, and W. Hong. Model-driven data acquisition in sensor networks. In *Proc. VLDB*, Toronto, ON, Canada, Aug. 2004.

[8] X. L. Dong, A. Y. Halevy, and C. Yu. Data integration with uncertainty. *VLDB Journal*, 18(2):469–500, 2009.

[9] R. Fagin, B. Kimelfeld, and P. Kolaitis. Probabilistic data exchange. In *Proc. ICDT*, Lausanne, Switzerland, Mar. 2010.

[10] E. Hung, L. Getoor, and V. S. Subrahmanian. PXML: A probabilistic semistructured data model and algebra. In *Proc. ICDE*, Bangalore, India, Mar. 2003.

[11] B. Kimelfeld, Y. Kosharovsky, and Y. Sagiv. Query efficiency in probabilistic XML models. In *Proc. SIGMOD*, Vancouver, BC, Canada, June 2008.

[12] B. Kimelfeld, Y. Kosharovsky, and Y. Sagiv. Query evaluation over probabilistic XML. *VLDB Journal*, 18(5):1117–1140, Oct. 2009.

[13] B. Kimelfeld and Y. Sagiv. Matching twigs in probabilistic XML. In *Proc. VLDB*, Vienna, Austria, Sept. 2007.

[14] C. Koch. MayBMS: A system for managing large uncertain and probabilistic databases. In C. Aggarwal, editor, *Managing and Mining Uncertain Data*. Springer-Verlag, 2009.

[15] J. Lafferty, A. McCallum, and F. Pereira. Conditional Random Fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. ICML*, Williamstown, NJ, USA, June 2001.

[16] A. Nierman and H. V. Jagadish. ProTDB: Probabilistic data in XML. In *Proc. VLDB*, Hong Kong, China, Aug. 2002.

[17] E. Rahm and P. A. Bernstein. A survey of approaches to automatic schema matching. *VLDB Journal*, 10(4):334–350, 2001.

[18] P. Senellart and S. Abiteboul. On the complexity of managing probabilistic XML data. In *Proc. PODS*, Beijing, China, June 2007.

[19] W3C. XQuery Update facility. `http://www.w3.org/TR/xquery-update-10/`, June 2009. Candidate Recommandation.

[20] J. Widom. Trio: A system for integrated management of data, accuracy, and lineage. In *Proc. CIDR*, Asilomar, CA, USA, Jan. 2005.

[21] XML::DB Initiative. XUpdate. `http://xmldb-org.sourceforge.net/xupdate/`, Sept. 2000. Working Draft.