Challenges for View-Based Query Answering over Probabilistic XML

Bogdan Cautis[1] and Evgeny Kharlamov[2]

[1] Institut Télécom; Télécom ParisTech, France
cautis@telecom-paristech.fr
[2] KRDB Research Centre, Free University of Bozen-Bolzano, Italy
kharlamov@inf.unibz.it

Abstract. This is the first and preliminary study on answering queries using views in a probabilistic XML setting. We formalize the problem and study it under the two possible semantics for XML query results: with node identifiers and in their absence. Accordingly, we consider rewrite plans that can exploit a single view, by means of compensation, and plans that can use multiple views, by means of intersection. Since in probabilistic settings queries return answers with probabilities, the problem of rewriting goes beyond the classical one of retrieving answers from views. For both semantics of XML queries, we show that, even if the XML answers can be retrieved, the computation of their probabilities might not be possible. We give restrictions that make probabilistic rewriting feasible in polynomial time, and present some initial hardness results for this problem.

1 Introduction

Uncertainty is ubiquitous in data and many applications must cope with this [1]: information extraction from the World Wide Web [2], automatic schema matching in data integration [3] are inherently imprecise. This uncertainty is sometimes represented as the *probability* that the data is correct, as with conditional random fields [4] in information extraction, or uncertain schema mappings in information integration [5]. In other cases, only *confidence* in the information is provided by the system, which can be seen after renormalization as an approximation of the probability. It is thus natural to manipulate such probabilistic information in a *probabilistic database management system* [6].

Recent work has proposed models for probabilistic data, both in the relational [7,8,9] and XML [10,11,12] settings. We focus here on the latter, which is particularly adapted for the Web. A number of studies on probabilistic XML have dealt with query answering for a variety of models and query languages [13,10,11,12,11]. At the same time, query optimization over probabilistic data has received little attention. In particular, the problem of answering queries using views, a key approach for optimization, has received no attention so far in both the relational and the semistructured settings. Probabilistic query evaluation could greatly benefit from such techniques, as it is often the case that computing probabilistic results is harder than in the deterministic setting.

Views over XML documents can be seen as fragments of data that may be available along with the nodes selected by the query. Over a p-document, the data fragments come

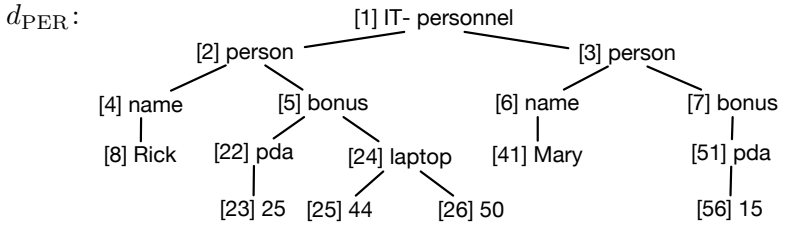

Fig. 1. Example document d_{PER}

together with their probability. In the general setting, we are given a document d and a set of view queries $v_1, \ldots, v_n$. Given a query q, the goal is to understand whether one can obtain $q(d)$, the answers of q over d, by accessing view results $v_1(d), \ldots, v_n(d)$ only. For XML data, the problem was studied under the two possible semantics for XML query results: with persistent node identifiers or in their absence. For the latter case, only rewrite plans that rely on a single view, by means of compensation, are possible. For the former, plans using multiple views, by means of intersection and compensation, are exploitable. We consider both settings and alternatives for rewritings.

We present a preliminary study of the problem of answering queries using views in the probabilistic XML setting. We formalize the problem in Section 3 and we give a preliminary study of it for the two possible settings: in the absence of node Ids, in Section 4, and in their presence, in Section 5. We show that, in the probabilistic setting, the problem of answering queries using views becomes more complex and it does not reduce to its deterministic version. The reason is that query results now involve not only data trees, but also their probabilities. Hence probabilities should also be retrieved from probabilistic view results, by means of a *probabilistic function* computing them.

Even for the simpler setting (without Ids), the existence of the probabilistic function is not guaranteed by the existence of a data-retrieving rewriting. We present examples of views and queries for which such a function does not exist. Based on a notion of probabilistic independence between queries, we also isolate a class of queries and views for which this function, when it exists, can be found and computed efficiently. For rewritings with intersection, we provide a sufficient condition (also based on probabilistic independence) that guarantees that the probabilities of query answers can be computed as a product-like formula over the probabilities of the views appearing in the intersection. We also present an NP-hardness result for deciding whether a selection of probabilistically independent views for a rewriting is possible.

2 Preliminaries

We briefly define here the data and query model. Details can be found in [12,14].

XML documents. We assume the existence of a countable set of labels $\mathcal{L}$ that subsumes both XML tags and values. We consider an XML document as an unranked, unordered rooted tree d modeled by a set of edges $\mathtt{edges}(d)$, a set of nodes $\mathtt{nodes}(d)$, a distinguished root node $\mathtt{root}(d)$ and a labeling function lbl, assigning to each node a label from $\mathcal{L}$. We assume that each node $n \in \mathtt{nodes}(d)$ has a unique *identifier*.

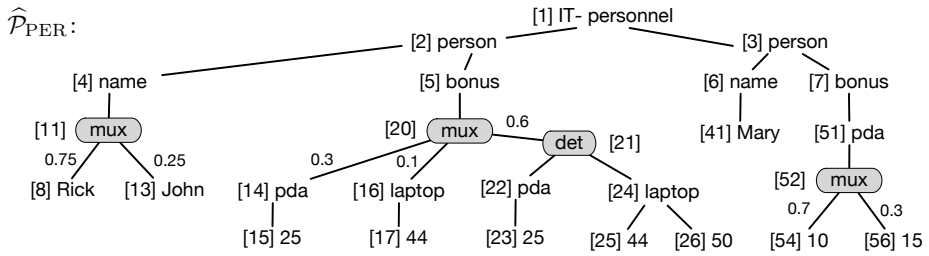

Fig. 2. Example p-document $\widehat{\mathcal{P}}_{\mathrm{PER}}$

Example 1. Consider the document d_{PER} in Figure 1 (where PER stands for personnel) describing the personnel of an IT department and the bonuses distributed for different projects. The document d_{PER} indicates that Rick worked under two projects (laptop and pda) and got bonuses of 44 and 50 in the former project and 25 in the latter one. Identifiers are written inside square brackets and labels are next to them, e.g., the node n_4 is labeled name, i.e., $\mathsf{lbl}(n_4) = \mathrm{name}$.

We define a *finite probability space of XML documents*, or *px-space* for short, as a pair $(\mathcal{D}, \Pr)$ with $\mathcal{D}$ a set of documents and $\Pr$ mapping every document d to a probability $\Pr(d)$ such that $\sum\{\Pr(d) \mid d \in \mathcal{D}\} = 1$.

Probabilistic documents. p-Documents give a general syntax for compactly representing px-spaces. Like a document, a p-document is a tree but with two kinds of nodes: *ordinary* nodes, which have labels and are the same as in documents, and *distributional*, which are used to define the probabilistic process for generating random documents. We consider three kinds of distributional nodes: *ind* (for *independent*), *mux* (for *mutually exclusive*), and *det* (for *deterministic*). Other kinds of distributional nodes are studied in [12], but *mux*, *det* alone are enough to represent all px-spaces as p-documents [12].

Definition 2. *A* p-document $\widehat{\mathcal{P}}$ *is an unranked, unordered tree with a set of edges* $\mathtt{edges}(\widehat{\mathcal{P}})$*, nodes* $\mathtt{nodes}(\widehat{\mathcal{P}})$*, the root node* $\mathtt{root}(\widehat{\mathcal{P}})$*, and a labeling function* lbl*, assigning to each node* v *a label from* $\mathcal{L} \cup \{\mathit{ind}(\Pr), \mathit{mux}(\Pr), \mathit{det}\}$*. If* $\mathsf{lbl}(v)$ *is* $\mathit{mux}(\Pr_v)$ *or* $\mathit{ind}(\Pr_v)$*, then* $\Pr_v$ *assigns to each child* v' *of* v *a probability* $\Pr_v(v')$*, and if* $\mathsf{lbl}(v) = \mathit{mux}(\Pr_v)$*, then also* $\sum_{v'} \Pr_v(v') \leq 1$*. We require leaves and the root to be* $\mathcal{L}$*-labeled.*

Example 3. Figure 2 shows a p-document $\widehat{\mathcal{P}}_{\mathrm{PER}}$ (where PER stands for personnel) that has *mux* and *det* distributional nodes, shown on gray background. Node n_{52} is a *mux* node with two children n_{54} and n_{56}, where $\mathsf{Pr}_{n_{52}}(n_{54}) = 0.7$ and $\mathsf{Pr}_{n_{52}}(n_{56}) = 0.3$.

A p-document $\widehat{\mathcal{P}}$ has as associated *semantics* a px-space $[\![\widehat{\mathcal{P}}]\!]$ defined by the following random process. Independently for each $\mathit{mux}(\Pr_v)$ (resp. $\mathit{ind}(\Pr_v)$) node, we select at most one (resp. some) of its children v' and delete all other children along with their descendants. We do not delete any of the children of *det* nodes. We then remove in turn each distributional node, connecting ordinary children of deleted distributional nodes with their lowest ordinary ancestors. The result of this process is a random document $\mathcal{P}$. The *probability* of $\mathcal{P}$, $\mathsf{Pr}(\mathcal{P})$, is the product of all *(i)* $\Pr_v(v')$ for each chosen child

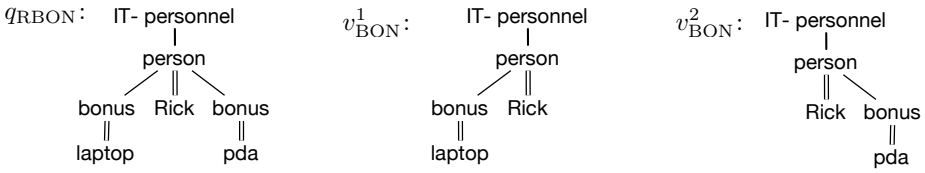

Fig. 3. Example TP query q_{RBON} and two TP views: v^1_{RBON} and v^2_{RBON}

v' of a *mux* or *ind* node v, *(ii)* $1 - \Pr_v(v')$ for each *not* chosen child v' of a *ind* node v, *(iii)* $1 - \sum_{v'} \Pr_v(v')$ for all children of each *mux* node v for which no children were chosen. Note that for any $\mathcal{P} \in [\![\widehat{\mathcal{P}}]\!]$ there is a unique way to generate it.

Example 4. Looking again at Figures 1 and 2, one can obtain the document d_{PER} from $\widehat{\mathcal{P}}_{\mathrm{PER}}$ by choosing: the left child of the *mux* node n_{11}, the right child of the *mux* node n_{20}, and the right one of child of the *mux* node n_{52}. The marginal probability of these choices (and the probability of d_{PER}), is $0.135 = 0.75 \times 0.6 \times 0.3$.

Tree-Pattern queries. The language of *tree-pattern queries* (TP) is roughly the subset of navigational XPath with child, descendant navigation, predicates, without wildcards.

Definition 5. *A* tree-pattern q *is a non-empty, unordered, unranked rooted tree, with a set of nodes* $\mathtt{nodes}(q)$ *labeled with symbols from* $\mathcal{L}$*, a node called the* output node $\mathtt{out}(q)$ *(i.e., tree-patterns are unary queries), and two types of edges:* child edges*, labeled by* / *and* descendant edges*, labeled by* //*. The root of* q *is denoted* $\mathtt{root}(q)$.

Due to space limitations, we will often write tree-patterns q in XPath notation [15], and denote this notation with $\mathtt{xpath}(q)$. We use $\mathsf{lbl}(q)$ as short notation for $\mathsf{lbl}(\mathtt{out}(q))$. We use the following graphical representation for tree-patterns: the main branch is the vertical path starting from the root, the output node is the last node of this path, and predicates are subtrees starting with side branches (see Figure 3).

Example 6. Consider the query q_{RBON} in Figure 3 (left) (where RBON stands for Rick's bonuses) asking whether Rick has received a bonus from the project laptop and a bonus from the project pda. In this representation, single lines denote child edges and double lines descendant edges. The other two queries v^1_{RBON} and v^2_{RBON} in Figure 3 (center and right) ask whether Rick has received a bonus on either of these projects. The output node in all the three queries is labeled with Rick.

The semantics of tree-patterns is given using embeddings. An *embedding* e of a TP query q into a document d is a function from $\mathtt{nodes}(q)$ to $\mathtt{nodes}(d)$ satisfying: *(i)* $e(\mathtt{root}(q)) = \mathtt{root}(d)$; *(ii)* for any $n \in \mathtt{nodes}(q)$, $\mathsf{lbl}(e(n)) = \mathsf{lbl}(n)$; *(iii)* for any /-edge (n_1, n_2) in q, $(e(n_1), e(n_2))$ is an edge in d; *(iv)* for any //-edge (n_1, n_2) in q, there is a path from $e(n_1)$ to $e(n_2)$ in d.

The *result* $q(d)$ of applying a tree-pattern q to a document d is the set:

$$q(d) := \{e(\mathtt{out}(q)) \mid e \text{ is an embedding of } q \text{ into } d\}$$

Example 7. Continuing with the three queries in Figure 3, they all return $\{n_8\}$ over the document d_{PER}, since Rick has got bonuses from both of the requested projects.

Intersections of tree-patterns. We consider in this paper the extension $\mathsf{TP}^\cap$ of TP with respect to intersection, which denotes *intersections of tree-pattern queries*.

$$\mathsf{TP}^\cap = \{q_1 \cap \cdots \cap q_k \mid k \in \mathbb{N}, q_i \in \mathsf{TP}, \text{ and } \mathsf{lbl}(\mathtt{root}(q_i)) = \mathsf{lbl}(\mathtt{root}(q_j)), \text{ and } \mathsf{lbl}(\mathtt{out}(q_i)) = \mathsf{lbl}(\mathtt{out}(q_j)) \text{ for } i,j \in \{1,\ldots,k\}\}.$$

The result of a $\mathsf{TP}^\cap$ query $q_1 \cap \cdots \cap q_k$ over a document d is the set of nodes $\bigcap_{i=1}^{k} q_i(d)$.

Query equivalence and containment. A pattern q_1 is *contained* in a pattern q_2, denoted $q_1 \sqsubseteq q_2$, if $q_1(d) \subseteq q_2(d)$ for every d. Also q_1 is *equivalent* to q_2, $q_1 \equiv q_2$, if $q_1 \sqsubseteq q_2$ and $q_2 \sqsubseteq q_1$. We discuss how to check containment of $\mathsf{TP}^\cap$ queries in Section 5. For TP queries, containment can be decided using containment mappings [16,17] which are similar to embeddings. Intuitively, a *containment mapping* from q_1 to q_2 is a function from $\mathtt{nodes}(q_1)$ to $\mathtt{nodes}(q_2)$ that respects the labels of nodes and maps any two nodes connected with $/$-edges to nodes connected with $/$-edges, while nodes connected with $//$-edges can be mapped to any connected nodes. Then for q_1 and q_2 in TP, $q_2 \sqsubseteq q_1$ iff there is a containment mapping from q_1 to q_2. Note that such a mapping can be computed in polynomial time. For example, observe that q_{RBON} is contained in both v^1_{RBON} and v^2_{RBON}, while none of the latter two is contained in each other.

Querying p-documents. Up to now, we have seen queries as functions over XML documents outputting sets of nodes. Over a p-document $\widehat{\mathcal{P}}$, a query q (TP or $\mathsf{TP}^\cap$) naturally yields a set of node-probability pairs (n,p), where n is a node of $\widehat{\mathcal{P}}$, and p is the probability that q can be embedded into a random document $\mathcal{P}$ of $\widehat{\mathcal{P}}$ with some e such that $e(\mathtt{out}(q)) = n$; this value will also be written as $\mathsf{Pr}(n \in q(\mathcal{P}))$. Formally:

$$q(\widehat{\mathcal{P}}) := \{(n,p) \mid \exists\, d \in [\![\widehat{\mathcal{P}}]\!] \colon n \in q(d) \text{ and } p = \sum\nolimits_{d \in [\![\widehat{\mathcal{P}}]\!] \colon n \in q(d)} \Pr(d)\}.$$

It is known [11] that TP queries can be evaluated over p-documents $\widehat{\mathcal{P}}$ in time polynomial in $|\widehat{\mathcal{P}}|$, that is, in *data-complexity*. The same holds for $\mathsf{TP}^\cap$ queries.

Example 8. Evaluation of q_{RBON} over $\widehat{\mathcal{P}}_{\mathrm{PER}}$ returns the node n_8 iff the left child of the node n_{11} and the right child of n_{20} are chosen. Hence, $q_{\mathrm{RBON}}(\widehat{\mathcal{P}}_{\mathrm{PER}}) = \{(n_8, 0.75 \times 0.6)\}$. Evaluation of v^1_{RBON} and v^2_{RBON} over $\widehat{\mathcal{P}}_{\mathrm{PER}}$ returns $\{(n_8, 0.75 \times (0.1 + 0.6))\}$ and $\{(n_8, 0.75 \times (0.3 + 0.6)\}$, respectively.

Further notations. We introduce now some additional terminology for TP queries, which will be used in Sections 4 and 5, and can be skipped until then.

The *main branch* $\mathtt{mb}(q)$ of q is the path from $\mathtt{root}(q)$ to $\mathtt{out}(q)$, and the *main branch nodes* of q, $\mathtt{mbn}(q)$, are the nodes of $\mathtt{mb}(q)$. A *prefix* q_p of q is any tree-pattern that can be build from q by setting $\mathtt{root}(q)$ as $\mathtt{root}(q_p)$, setting some node $n \in \mathtt{mbn}(q)$ as $\mathtt{out}(q_p)$, and removing from q all $\mathtt{mbn}(q)$ nodes below n along with their descendants. A *suffix* q_s of q is any subtree of q rooted at a node of $\mathtt{mbn}(q)$. The *rank* of a main branch node is the distance from it to the root, i.e., the rank of $\mathtt{root}(q)$ is 1 and of $\mathtt{out}(q)$ is $|\mathtt{mbn}(q)|$. The suffix of q rooted at the node of rank k is denoted q^k. For any rank k, *cut*(q,k) denotes the prefix of q with k main branch nodes.

3 Problem Definition

We assume an infinite set of *view names* $\mathcal{V}$ disjoint from the set of labels $\mathcal{L}$. By a *view* v we denote a tree-pattern query (that *defines* the view) together with its *name* $v \in \mathcal{V}$.

Deterministic view-based rewriting. Let d be a document and v a view. A *(deterministic) view extension* of v over d, denoted d_v, is a *document* obtained by connecting to a root node labeled $doc(v)$ all the documents from the set $\{d' \mid d' \subseteq d \text{ and } \texttt{root}(d') \in v(d)\}$. Such a document can be queried by TP-queries of the form $doc(v)/\mathsf{lbl}(v)/\ldots$. If V is a set of views defined over a document d, then $D_V^d = \{d_v \mid v \in V\}$. Let q be a query in $\mathcal{Q} \in \{\text{TP}, \text{TP}^\cap\}$ that may use $doc(v)/\mathsf{lbl}(v)$ for $v \in V$, then $unfold_V(q)$ is the query in $\mathcal{Q}$ obtained by replacing in q each $doc(v)/\mathsf{lbl}(v)$ with the definition of v.

Example 9. Two views v^1_{RBON} and v^2_{RBON} are in Figure 3. Their extensions over d_{PER} are, respectively, documents $(d_{\text{PER}})_{v^1_{\text{RBON}}}$ and $(d_{\text{PER}})_{v^2_{\text{RBON}}}$ each with two nodes: the root labeled, respectively, with $doc(v^1_{\text{RBON}})$ and $doc(v^2_{\text{RBON}})$, and with one child n_8 labeled Rick in both cases. Let v be v^2_{RBON} without the node labeled Rick and with the output node person. Then $(d_{\text{PER}})_v$ has the root labeled $doc(v)$ to which two subdocuments of d_{PER} are connected: a subdocument rooted at n_2 and one at n_3.

Let d be a document, q a TP-query, V a set of TP-views, and $\mathcal{Q} \in \{\text{TP}, \text{TP}^\cap\}$. In the deterministic setting the problem of query answering using views is to find an alternative query plan q_r in $\mathcal{Q}$, called a *rewriting*, that can be used to answer q. Formally, a $\mathcal{Q}$*-rewriting* q_r of q using V is a query $q_r \in \mathcal{Q}$ such that for every document d it holds that $q_r(D_V^d) = q(d)$. Clearly, this implies that $unfold_V(q_r) \equiv q$.

The two alternatives, TP-*rewritings* and TP$^\cap$-*rewritings*, are respectively motivated by the two possible interpretations of XML query results. In an XML document, nodes have unique Ids used by internal operators (selections, unions, joins, etc.) to manipulate data during query evaluation. Queries can then either *(i)* output fresh Ids for the nodes of the result, or *(ii)* expose (preserve) in the result the original Ids from the document. The former case corresponds to what is called *the copy semantics*, under which the Ids of any document in D_V^d are disjoint from those of d and from those of any other document in D_V^d. Since one cannot know if nodes from results of different views are in fact copies of the same node in d, the only possible rewritings are those that access a single document from D_V^d and maybe navigate inside it. A rewriting $q_r \in$ TP will thus be of the form $doc(v)/\mathsf{lbl}(v)[p_1]/p_2$ or $doc(v)/\mathsf{lbl}(v)[p_1]//p_2$, where $v \in V$ and the (possibly empty) TP-queries p_1, p_2 represent the *compensation* of v. In the latter case, every document in D_V^d preserves the original Ids, which will identify nodes across different documents in D_V^d. One can thus formulate and exploit more complex rewritings, as node Ids can be used to intersect (join by Id) results of different views over the same input data d. TP$^\cap$*-rewritings* q_r extend TP-rewritings in that they can access several D_V^d documents at once, by first navigating in individual documents and then intersecting the result. Thus the form of TP$^\cap$-rewritings is $\bigcap_{i,j} u_{ij}$, where each u_{ij} is a TP-rewriting.

Probabilistic view-based rewriting. We generalize the definition of view extension to the probabilistic case: $\widehat{\mathcal{P}}_v$ is a p-document rooted at a node labeled $doc(v)$ whose contents is constructed as follows: (i) plug an unique *ind*-child below $\texttt{root}(\widehat{\mathcal{P}}_v)$, (ii) for

each pair (α, β) in the set $\{(\widehat{\mathcal{P}}', p) \mid \widehat{\mathcal{P}}' \subseteq \widehat{\mathcal{P}}$ and $(\texttt{root}(\widehat{\mathcal{P}}'), p) \in q(\widehat{\mathcal{P}})\}$, add α as subtree of this *ind*-node with the probability β. A set of p-documents $D_V^{\widehat{\mathcal{P}}}$ for the set of views V and unfolding of a query over $D_V^{\widehat{\mathcal{P}}}$ is defined as in the deterministic case.

Example 10. Continuing with Example 9, extensions $(\widehat{\mathcal{P}}_{\text{PER}})_{v^1_{\text{RBON}}}$ $(\widehat{\mathcal{P}}_{\text{PER}})_{v^2_{\text{RBON}}}$ of the views over $\widehat{\mathcal{P}}_{\text{PER}}$ are p-documents with three nodes: the roots are labeled respectively $doc(v^1_{\text{RBON}})$ and $doc(v^2_{\text{RBON}})$, with one child labeled *ind*, that in turns has one child with the id n_8 labeled Rick. The edge between the *ind*-node and its child is labeled 0.75 in both cases. The extension $(\widehat{\mathcal{P}}_{\text{PER}})_v$ has the root labeled v, with one child labeled *ind*, and two p-subdocuments of $\widehat{\mathcal{P}}_{\text{PER}}$ rooted under the ind node: one is the p-subdocument rooted at n_2 and the other one rooted at n_3. Probabilities on the edges to n_2 and n_3 are 1.

Query answering using views in the probabilistic setting is more involved than in the deterministic one, since $q(\widehat{\mathcal{P}})$ is a set of node-probability pairs. Therefore, rewrite plans should deal with two sub-problems: *(i)* to find a query in terms of views, that retrieves the nodes N of $q(\widehat{\mathcal{P}})$ (this corresponds to deterministic rewriting plans) and *(ii)* to compute the probabilities for the nodes in N, using probabilities from $\mathcal{D}_V^{\widehat{\mathcal{P}}}$. Both sub-problems require algorithms accessing p-documents $\mathcal{D}_V^{\widehat{\mathcal{P}}}$ only. More formally:

Definition 11. *Let q be a* TP *query, V be a set of* TP *views and $\mathcal{Q} \in \{\text{TP}, \text{TP}^\cap\}$. A* probabilistic $\mathcal{Q}$-rewriting $Q_r = (q_r, f_r)$ *of q using V is a pair of*

(i) a deterministic $\mathcal{Q}$-rewriting q_r of q using V, and

(ii) a probability function f_r *such that for every p-documents $\widehat{\mathcal{P}}$ and every node n of $\widehat{\mathcal{P}}$ it holds that $f_r(n, D_V^{\widehat{\mathcal{P}}}) = \mathsf{Pr}(n \in q(\mathcal{P}))$.*

When $D_V^{\widehat{\mathcal{P}}}$ is clear from the context we will use $f_r(n)$ as short notation for $f_r(n, D_V^{\widehat{\mathcal{P}}})$.

For given q and V, a *probabilistic rewriting problem* is to find Q_r. The main challenge in solving this problem is to construct a probability function f_r that, by definition, has access only to the p-documents in $D_V^{\widehat{\mathcal{P}}}$. In Sections 4 and 5 we respectively show that this is not always possible for TP and TP$^\cap$-rewritings.

4 TP-rewrite Plans

Here we discuss when probabilistic TP-rewrite plans do not exist and present cases for which they do exist and can be computed in polynomial time.

We first introduce some auxiliary notation. By d^n we denote the subdocument of d rooted at n. We denote the p-subdocument of $\widehat{\mathcal{P}}$ rooted at a node n as $\widehat{\mathcal{P}}^n$. For TP queries q_1 and q_2, *compensation* of q_1 with q_2, denoted $\texttt{comp}(q_1, q_2)$, is a TP-query obtained by deleting the first symbol from $\texttt{xpath}(q_2)$ and concatenating the rest to $\texttt{xpath}(q_1)$. For instance, the result of compensating $q_1 = a/b$ with $q_2 = b[c][d]/e$ is the concatenation of a/b and $[c][d]/e$, i.e., $\texttt{comp}(q_1, q_2) = a/b[c][d]/e$. Intuitively, view's compensation brings further navigation over the view's result $\widehat{\mathcal{P}}_v$, and a rewrite plan will be of the form $q_r = \texttt{comp}(doc(v)/\mathsf{lbl}(v), q^k)$ such that its unfolding $\texttt{comp}(v, q^k)$ is equivalent to q. Note that q_r is over $\widehat{\mathcal{P}}_v$, its unfolding is over $\widehat{\mathcal{P}}$ while they yield the same result.

We remind the reader the main result for deterministic compensations [18]:

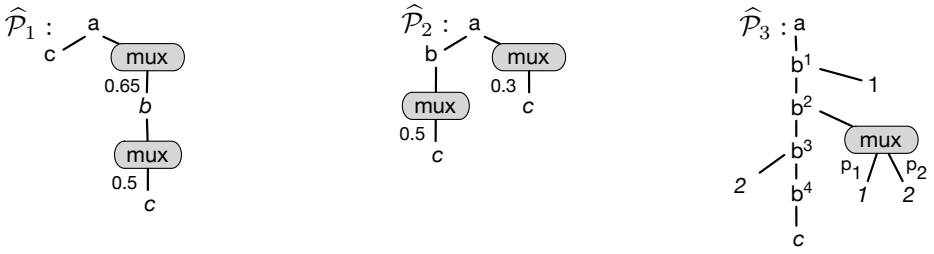

Fig. 4. p-Documents to show non-existence of TP-rewritings (Examples 14 and 16)

Fact 12. *Let q and V be* TP*-queries. Then there exists a deterministic* TP*-rewriting of q over V if and only if there is $v \in V$ and $k \in \mathbb{N}$ such that $\mathtt{comp}(v, q^k) \equiv q$.*

This criterion can be verified in polynomial time [18]. Fact 12 says that, using just one view v from V, we can find all the nodes $n \in q(d)$ by querying d_v with q^k, i.e., the data in d_v suffices to extract all such n. This naturally extends to the probabilistic setting: we can find all the nodes $n \in q(\widehat{\mathcal{P}})$ by querying $\widehat{\mathcal{P}}_v$ with q^k, i.e., the data in $\widehat{\mathcal{P}}_v$ it suffices to extract all ns. Note that n is in the query result $q(\mathcal{P})$ iff $\mathsf{Pr}(n \in q(\mathcal{P})) > 0$.

Proposition 13. *Let q and v be* TP*-queries and $k \in \mathbb{N}$. Let $q_r = \mathtt{comp}(doc(v)/\mathsf{lbl}(v), q^k)$ be a deterministic* TP*-rewriting of q using v. Then for every p-document $\widehat{\mathcal{P}}$ it holds*

$$\mathsf{Pr}(n \in q(\mathcal{P})) > 0 \quad \textit{if and only if} \quad \mathsf{Pr}(n \in q_r(\mathcal{P}_v)) > 0.$$

4.1 Nonexistence of TP-rewrite Plans

Is information in $\widehat{\mathcal{P}}_v$ also sufficient to extract the probabilities $\mathsf{Pr}(n \in q(\mathcal{P}))$ for nodes $n \in q(\mathcal{P})$? It turns out that the answer is negative. There are q and v for which a deterministic rewriting q_r exists but not the probabilistic one, i.e. the function f_r such that for every $\widehat{\mathcal{P}}$ it holds that $f_r(n) = \mathsf{Pr}(n \in q(\mathcal{P}))$ does not exist. Thus the probabilistic rewriting problem crucially different from the deterministic one. We now present two example that will give insides on this phenomenon.

Example 14. Consider the query $q = a/b[c]$ and the view $v = a[.//c]/b$. We now show that there is no probabilistic rewriting (q_r, f_r) for q over $\{v\}$. One can see that $\mathtt{comp}(v, q^2) = a[.//c]/b[c]$ is equivalent to q, hence, $q_r = \mathtt{comp}(doc(v)/\mathsf{lbl}(v), q^2)$. Consider now two p-documents $\widehat{\mathcal{P}}_1$ and $\widehat{\mathcal{P}}_2$ from Figure 4. Clearly, $\mathsf{Pr}(b \in q(\mathcal{P}_1)) = 0.65 \times 0.5$ and $\mathsf{Pr}(b \in q(\mathcal{P}_2)) = 0.5$, and these probabilities are different. The function f_r should compute the first probability 0.325 on a p-document $(\widehat{\mathcal{P}}_1)_v$ and 0.5 on $(\widehat{\mathcal{P}}_2)_v$, hence f_r should distinguish these p-documents. While one can see that these p-documents are *indistinguishable* by v: $(\widehat{\mathcal{P}}_1)_v = (\widehat{\mathcal{P}}_2)_v$. Hence, f_r does not exist.

The problem raised in this example comes from the fact that in the unfolding of the rewriting $a[.//c]//b[c]$ the predicate $[.//c]$ coming from the view (i.e. located above the b-labeled node $\mathtt{out}(q)$) and the predicate $[c]$ coming from the compensation, (i.e. located below $\mathtt{out}(q)$) may interact. Where the interaction is of the following kind: there is a document, e.g., d with the root a that has one child b, which in tern has one child c.

that satisfies $a[.//c]//b[c]$ but both c-nodes of the query should be mapped to the same c-node of d. In other words, the existence of a match for one predicate depends on the (non-)existence of a match of the other. We now introduce a condition of probabilistic independence that prevents such an interaction. This condition will be further used for both TP and TP$^\cap$-rewritings.

TP-queries q_1 and q_2 are *independent*, denoted $q_1 \perp q_2$, if for every $\widehat{\mathcal{P}}$ and $n \in \widehat{\mathcal{P}}$:

$$\mathsf{Pr}(n \in (q_1 \cap q_2)(\mathcal{P})) = [\mathsf{Pr}(n \in q_1(\mathcal{P})) \times \mathsf{Pr}(n \in q_2(\mathcal{P}))] \div \mathsf{Pr}(n \in \mathcal{P}).$$

Clearly, the view $a[.//c]//b$ and the compensation $b[c]$ from Example 14 are probabilistically dependent. Deciding probabilistic dependency is tractable:

Proposition 15. *For* TP*-queries the independence $q_1 \perp q_2$ is decidable in polynomial time.*

Observe that probabilistic independence between a view and its compensation in a deterministic rewriting does not guarantee existence of a probabilistic rewriting.

Example 16. Consider $q = a//b[1]/b[2]/b//c$ and $v = a//b[1]/b[2]/b$. Clearly, q^2 is a compensation for v and $q_r = \mathtt{comp}(v, q^2)$ is a deterministic TP-rewriting for q and v. We now present two p-documents $\widehat{\mathcal{P}}_3$ and $\widehat{\mathcal{P}}_4$ that show the non-existence of a probability function f_r, such that (q_r, f_r) is a probabilistic TP-rewriting for q and v. Consider $\widehat{\mathcal{P}}_3$ from Figure 4. where the upper index ^{i} on b indicates the i-th occurrence of a node labeled b in $\widehat{\mathcal{P}}_3$. Clearly, $(\widehat{\mathcal{P}}_3)_v$ is a p-document with the root that has one *ind* child, under which two p-documents: $\widehat{\mathcal{P}}_{b^{(3)}}$ with probability p_2 and $\widehat{\mathcal{P}}_{b^{(4)}}$ with p_1 are rooted. Consider now $\widehat{\mathcal{P}}_4$ that is different from $\widehat{\mathcal{P}}_3$ in that it has an *ind*-node instead of the *mux*-node. Clearly, $(\widehat{\mathcal{P}}_3)_v = (\widehat{\mathcal{P}}_4)_v$. Note that both $\widehat{\mathcal{P}}_{b^{(3)}}$ and $\widehat{\mathcal{P}}_{b^{(4)}}$ are deterministic documents, and in both $(\widehat{\mathcal{P}}_i)_v$ there is no information on whether p_1 and p_2 are coming from the same distributional node or not, and if they come from the same node, then there is no information on what type of this node is. At the same time, $\mathsf{Pr}(c \in q(\widehat{\mathcal{P}}_3)) = p_1 + p_2$, while $\mathsf{Pr}(c \in q(\widehat{\mathcal{P}}_4)) = p_1 + p_2 - p_1 \times p_2$. That is, the probabilities are different and the result depends on the kind of the probabilistic relations in which p_1 and p_2 are involved, i.e., on whether they are for associated with the children of a *mux* or *ind* distributional node. If a probability computation function f_r for q_r exists then it should be able compute the latter two different probabilities from the same views $v(\widehat{\mathcal{P}}_3)$ and $v(\widehat{\mathcal{P}}_4)$, which is impossible since they have no information on the relationship between p_1 and p_2.

4.2 Simple Queries and Restricted Compensations

We now provide a class of queries and a class of compensated queries for which deciding existence of probability rewriting is based on probabilistic independence.

A query q is *simple* if either its main branch has /-edges only, or all the nodes in $\mathtt{mbn}(q) \setminus \mathtt{out}(q)$ reachable from (the first occurrence of) a //-edge have no predicates. Clearly, a query is not simple if it is of the form $\ldots // \ldots [\ldots] \ldots$. The compensation of a view v with c is *restricted* if either v is simple, or $\mathtt{mb}(c)$ has no //-edges.

Let n' be the highest ancestor-or-self node of n occurring in $v(\widehat{\mathcal{P}})$ and let $k = |\mathtt{mb}(v)|$. We show that under some conditions on q_r and v, the probability $\mathsf{Pr}(n \in q(\mathcal{P}))$

INPUT : TP query q and views V
OUTPUT: Set of TP-rewritings R

$R := \emptyset$,
Prefs $:= \{(q_1, v_i) \mid v_i \in V, q_1$ *is a lossless prefix of* $q, \mathtt{mb}(q_1) \equiv \mathtt{mb}(v_i), q_1 \sqsubseteq v_i\}$;
for *each* $(q_1, v_i) \in$ *Prefs* **do**
  $k := |\mathtt{mb}(q_1)| = |\mathtt{mb}(v_i)|$
  if $\mathtt{comp}(v_i, q^k) \equiv q$ *and restricted* **then**
    $v_i' := \mathtt{comp}(\mathtt{mb}(v_i), v_i^k)$
      // v_i w/o predicates of nodes at rank $1, \ldots, k-1$
    $v_i'' := \mathtt{comp}(\mathtt{cut}(v_i, k-1), \mathtt{mb}(v_i)^{k-1})$
      // v_i w/o predicates of node at rank k
    $q' := \mathtt{comp}(\mathtt{cut}(\mathtt{mb}(q), k), q^k)$
      // q w/o predicates of nodes at rank $1, \ldots, k-1$
    if $v_i' \perp v_i''$ *and* $v_i'' \perp q'$ **then** $R := R \cup \{\mathtt{comp}(doc(v)/\mathsf{lbl}(v), q^k)\}$

Algorithm 1: TPrewrite for finding TP-rewritings q_r for which f_r are as in Eq. 1

that a node n occurs in q's result is the probability $\mathsf{Pr}(n \in q_r(\mathcal{P}_v))$ that n can be found by q_r in $q_r(\widehat{\mathcal{P}}_v)$, divided by the probability $\mathsf{Pr}(n' \in v^k(\mathcal{P}_v^{n'}))$ that n' verifies the predicates of v found on its output node $\mathtt{out}(v)$. The following theorem is the main result of this section, where for $k = |\mathtt{mbn}(v)|$ the query $v' = \mathtt{comp}(\mathtt{mb}(v), v^k)$ is v without all predicates of (main branch) nodes of rank in $\{1, \ldots, k-1\}$, the query $v'' = \mathtt{comp}(\mathtt{cut}(v, k-1), \mathtt{mb}(v)^{k-1})$ is v without all predicates of the node at rank k (i.e., the output node), and $q' = \mathtt{comp}(\mathtt{cut}(\mathtt{mb}(q), k), q^k)$ is q without all predicates of main branch nodes of rank in $\{1, \ldots, k-1\}$.

Theorem 17. *Let v be a* TP*-view, q a* TP*-query, and $k = |\mathtt{mb}(v)|$. Let q_r be a restricted* TP*-rewriting of q over v. If $v' \perp v''$ and $v'' \perp q'$ hold, then for every $n \in q(\mathcal{P})$ and its highest ancestor-or-self node $n' \in v(\mathcal{P})$:*

$$\mathsf{Pr}(n \in q(\mathcal{P})) = \mathsf{Pr}(n \in q_r(\mathcal{P}_v)) \div \mathsf{Pr}(n' \in v^k(\mathcal{P}_v^{n'})). \quad (1)$$

We summarize this section with a polynomial time algorithm TPrewrite (see Algorithm 1), that takes an the input a TP query q and a set of views V and returns all possible TP-rewritings q_r, for which the probability functions f_r are as in Equation 1. In the algorithm we use so-called lossless prefixes: q_1 is a *lossless prefix* of q, if q_1 is a tree-pattern obtained from q by setting $\mathtt{out}(q_1)$ as some node of $\mathtt{mbn}(q)$.

5 TP$^\cap$-rewrite Plans

First we discuss how to decide equivalence between TP and TP$^\cap$ queries and then provide a restriction on views for which probability functions f_r exist and tractable.

5.1 Equivalence and containment for TP$^\cap$

Since Q in TP$^\cap$ is a rewriting for q in TP iff $unfold_V(Q) \equiv q$, deciding whether a TP query q is equivalent to a TP$^\cap$ query Q is crucial in our setting. It is known [14] that one

can rely on mappings to decide the equivalence: if q is first equivalently reformulated into the union of TP queries $\cup_i q_i$, called its possible *interleavings*, and can be exponentially large in $|Q|$. Interleavings capture all the possible ways to order or coalesce the main branch nodes of queries participating in the intersection. Testing $q \equiv Q$ is coNP-hard and boils down to testing $q \equiv \cup_i q_i$, which in turn boils down to testing: if for *some* j: $q \sqsubseteq q_j$, and if for *all* i: $q_i \sqsubseteq q$. We conclude:

Corollary 18. *Deciding existence of a probabilistic* TP$^\cap$*-rewriting for a* TP *query* q *and* TP *views* V *views is* coNP*-hard.*

The equivalence problem was however shown in [14] to be in PTIME when q belongs to a restricted fragment of TP, called *extended skeletons*. As our focus in this paper is on polynomial time algorithms for view-based rewriting, it is thus natural to ask if view-based rewriting over probabilistic data remains tractable when input queries are extended skeletons and expose node Ids, while views are general TP queries.

5.2 Computing Probability Function f_r for Mutually Independent TP$^\cap$ Views

We start with the assumption that a deterministic rewriting q_r has been found. Without loss of generality, let us assume that q_r consists only of intersected views (no compensations of views). Let $v_1, \ldots, v_k$ be the TP views of q_r. Towards building the probability component of the rewrite plan, f_r, we give some intuition first: for a given node $n \in q(\mathcal{P})$, since each view v_i gives a probability, $n \in v_i(\mathcal{P})$, and since we are interested in the probability of the intersection thereof, we might be tempted to try what is arguably the most intuitive definition for f_r, the one which would simply combine by *multiplication* the probabilities $\mathsf{Pr}(n \in v_i(\mathcal{P}))$. But there are two issues with this straightforward f_r candidate.

Regarding the first issue, a basic principle in probability theory is that the joint probability of several events equals the product of the individual probabilities only when these events are *independent*. For instance, for a given node $n \in \mathcal{P}$, are $\mathsf{Pr}(n \in a[1](\mathcal{P})]$ and $\mathsf{Pr}(n \in a[2](\mathcal{P})]$ independent? The answer is obviously negative, and we can readily construct $\widehat{\mathcal{P}}$ instances over which both values are non-zero but the probability of $\mathsf{Pr}(n \in a[1][2](\mathcal{P})]) = 0$. We have introduced the notion of query independence in the previous section, denoted $v_i \perp v_j$, which guarantees that the existence of some embedding of v_i in a given document does not depend on the existence (or non-existence) of some embedding of v_j in this document. We will see now that for pairwise independent views a function f_r is based on multiplication of these views' probabilities.

Regarding the second issue, for each node n that appears in $q(\widehat{\mathcal{P}})$ and, consequently, appears in each $v_1(\widehat{\mathcal{P}}), \ldots, v_k(\widehat{\mathcal{P}})$, we have k probability values $\mathsf{Pr}(n \in v_i(\mathcal{P}))$. Furthermore, each value $\mathsf{Pr}(n \in v_i(\mathcal{P}))$ can be seen as the product of two distinct terms:

(i) the probability of n appearing in a possible world of $\widehat{\mathcal{P}}$, denoted $\mathsf{Pr}(n \in \mathcal{P})$,

(ii) the probability of n being *selected* by v_i in a possible world *in which* n *is known to appear*, denoted in the following $\mathsf{Pr}(n \in v_i(\mathcal{P}) \mid n \in \mathtt{nodes}(\mathcal{P}))$.

Note that the first term is independent of any particular view as it only depends on the document itself We can thus write for each v_i and each pair $(n, p_i) \in v_i(\mathcal{P})$ that

$$p_i = \mathsf{Pr}(n \in \mathcal{P}) \times \mathsf{Pr}(n \in v_i(\mathcal{P}) \mid n \in \mathtt{nodes}(\mathcal{P})).$$

Given a deterministic rewriting q_r of q formed by pairwise independent views $v_1, \ldots, v_k$, for a node $n \in q(\mathcal{P})$), we would thus have as the overall product the following:

$$\prod_i \mathsf{Pr}(n \in v_i(\mathcal{P})) = \mathsf{Pr}(n \in \mathcal{P})^k \times \prod_i \mathsf{Pr}(n \in v_i(\mathcal{P}) \mid n \in \mathtt{nodes}(\mathcal{P})). \quad (2)$$

Observe that in Equation 2 we account for the probability $\mathsf{Pr}(n \in \mathcal{P})$ too many times, once for each view that participates in the rewrite plan. But we should instead account for it only once. Hence, by dividing Equation 2 with $\mathsf{Pr}(n \in \mathcal{P})^{k-1}$, we obtain f_r:

$$f_r(n) := \mathsf{Pr}(n \in \mathcal{P}) \times \prod_i \mathsf{Pr}(n \in v_i(\mathcal{P}) \mid n \in \mathtt{nodes}(\mathcal{P})). \quad (3)$$

Each independent view v_i gives us $\mathsf{Pr}(n \in v_i(\mathcal{P}) \mid n \in \mathtt{nodes}(\mathcal{P}))$, while there is now one missing ingredient in Equation 3: $\mathsf{Pr}(n \in \mathcal{P})$. We can compute this value only if there is a view $v_i \in V$ subsuming $\mathtt{mb}(q)$, i.e., $\mathtt{mb}(q) \sqsubseteq v_i$. Summing up we conclude:

Theorem 19. *Let q be a* TP*-query, V a set of pairwise independent* TP*-views s.t. there is $v \in V$ satisfying $\mathtt{mb}(q) \sqsubseteq v$. Let q_r be a* TP$^\cap$*-rewriting of q over V. Then (q_r, f_r) with the f_r as in Equation 3 is a probabilistic* TP$^\cap$*-rewriting of q over V.*

The next theorem shows that for //-free q and V' it is hard to decide the existence of $V \subseteq V'$ of pairwise independent views (by reduction from k-dimensional perfect matching). Thus, deciding whether the result of Theorem 19 is applicable even for //-free TP queries and views is intractable. This also implies that for extended skeletons it is hard to find TP$^\cap$-rewritings using the probability function as in Equation 3.

Theorem 20. *Let q and views V be both of* TP *and without //-edges, deciding whether a* TP$^\cap$*-rewriting of q using only pairwise independent views from V exists is* NP*-hard.*

6 Conclusion

This is the first study on answering queries using views over probabilistic XML. The main challenge of this problem is to find probability-retrieving functions that can access only view results, while able to compute the probabilities of XML answers. So far we studied two cases of TP and TP$^\cap$-rewritings where these functions exist under the assumption of probabilistic independence for queries and views. In the TP case, our setting allows for polynomial time query answering in the size of both data and query. In the TP$^\cap$ case, our setting allows for polynomial time query answering in the size of data, while it is intractable in the number of views. Recall that (direct) query answering for probabilistic XML model considered here is also polynomial in data and intractable in query complexity [11]. Moreover, query answering techniques of [11] are based on an expensive dynamic programming approach. At the same time, in our TP$^\cap$ setting, probability computation is done by means of the f_r function (Equation 3) that requires to compute probabilities that a node n occurs in p-documents. This computation is both conceptually and computationally easier than dynamic programming. It requires to collect the probabilities occurring on the way from the root of a p-document to the

node n, and to multiply them. Hence, a practical implication of our study is that query answering using TP$^\cap$-views in our restricted setting should be more efficient than direct query evaluation $q(\widehat{\mathcal{P}})$. As for the TP setting, reasoning over the views is tractable, while view-based query answering may require navigation in a view result $\widehat{\mathcal{P}}_v$, which in practice is often considerably smaller than $\widehat{\mathcal{P}}$. As for the future work, one extension is to broaden the setting and to understand how one can cope with probabilistic dependences in queries and views. Another extension concerns data: p-documents studied in this paper have local probabilistic dependences, while there are models allowing for more complex probabilistic interactions between remote fragments of data [19]. For these types of XML data, query answering is intractable and we would like to see under which conditions we can gain tractability by relying on views.

Acknowledgements. We are grateful to the anonymous reviewers for their comments. The second author is supported by the ERC FP7 grant Webdam (agreement n. 226513).

References

1. Kharlamov, E., Nutt, W., Senellart, P.: Value joins are expensive over (probabilistic) XML. In: Proc. LID, Uppsala, Sweden (2011)
2. Chang, C.H., Kayed, M., Girgis, M.R., Shaalan, K.F.: A survey of Web information extraction systems. IEEE TKDE **18**(10) (2006)
3. Rahm, E., Bernstein, P.: A survey of approaches to automatic schema matching. VLDBJ '01
4. Lafferty, J., McCallum, A., Pereira, F.: Conditional Random Fields: Probabilistic models for segmenting and labeling sequence data. In: Proc. ICML. (2001)
5. Dong, X.L., Halevy, A.Y., Yu, C.: Data integration with uncertainty. VLDBJ **18**(2) (2009)
6. Dalvi, N., Ré, C., Suciu, D.: Probabilistic databases: Diamonds in the dirt. CACM '09
7. Widom, J.: Trio: A system for integrated management of data, accuracy, and lineage. In: Proc. CIDR, Online Proceedings (2005) 262–276
8. Dalvi, N., Suciu, D.: The dichotomy of conjunctive queries on probabilistic structures. In: Proc. PODS. (2007)
9. Koch, C.: MayBMS: A system for managing large uncertain and probabilistic databases. In Aggarwal, C., ed.: Managing and Mining Uncertain Data. Springer, New York, NY (2009)
10. Nierman, A., Jagadish, H.V.: ProTDB: Probabilistic data in XML. In: Proc. VLDB. (2002)
11. Kimelfeld, B., Kosharovsky, Y., Sagiv, Y.: Query evaluation over probabilistic XML. VLDBJ **18**(5) (2009) 1117–1140
12. Abiteboul, S., Kimelfeld, B., Sagiv, Y., Senellart, P.: On the expressiveness of probabilistic XML models. VLDBJ **18**(5) (2009) 1041–1064
13. Abiteboul, S., Chan, T.H.H., Kharlamov, E., Nutt, W., Senellart, P.: Aggregate queries for discrete and continuous probabilistic XML. In: Proc. ICDT. (2010)
14. Cautis, B., Deutsch, A., Onose, N., Vassalos, V.: Querying XML data sources that export very large sets of views. TODS (2011)
15. Benedikt, M., Koch, C.: XPath leashed. ACM Comput. Surv. **41**(1) (2008)
16. Amer-Yahia, S., Cho, S., Lakshmanan, L.V.S., Srivastava, D.: Tree pattern query minimization. VLDBJ **11**(4) (2002) 315–331
17. Miklau, G., Suciu, D.: Containment and equivalence for a fragment of XPath. J. ACM '04
18. Xu, W., Özsoyoglu, Z.: Rewriting XPath queries using materialized views. In: Proc. VLDB. (2005) 121–132
19. Senellart, P., Abiteboul, S.: On the complexity of managing probabilistic XML data. In: Proc. PODS. (2007) 283–292