Capturing Continuous Data and Answering Aggregate Queries in Probabilistic XML

SERGE ABITEBOUL, INRIA Saclay – Île-de-France & LSV, ENS Cachan
T.-H. HUBERT CHAN, The University of Hong Kong
EVGENY KHARLAMOV, Free University of Bozen-Bolzano & INRIA Saclay – Île-de-France
WERNER NUTT, Free University of Bozen-Bolzano
PIERRE SENELLART, Institut Télécom; Télécom ParisTech; CNRS LTCI

Sources of data uncertainty and imprecision are numerous. A way to handle this uncertainty is to associate probabilistic annotations to data. Many such probabilistic database models have been proposed, both in the relational and in the semi-structured setting. The latter is particularly well adapted to the management of uncertain data coming from a variety of automatic processes. An important problem, in the context of probabilistic XML databases, is that of answering aggregate queries (count, sum, avg, etc.), which has received limited attention so far. In a model unifying the various (discrete) semi-structured probabilistic models studied up to now, we present algorithms to compute the distribution of the aggregation values (exploiting some regularity properties of the aggregate functions) and probabilistic moments (especially expectation and variance) of this distribution. We also prove the intractability of some of these problems and investigate approximation techniques. We finally extend the discrete model to a continuous one, in order to take into account continuous data values, such as measurements from sensor networks, and extend our algorithms and complexity results to the continuous case.

25

1. INTRODUCTION

The (HTML or XML) Web is an important source of uncertain data, for instance, generated by imprecise automatic tasks such as information extraction. A natural way to

model this uncertainty is to annotate semistructured data with probabilities. A number of studies consider queries over such imprecise hierarchical information[Nierman and Jagadish 2002; Hung et al. 2003; 2007; van Keulen et al. 2005; Abiteboul and Senellart 2006; Senellart and Abiteboul 2007; Kimelfeld and Sagiv 2007; Kimelfeld et al. 2008]. An essential aspect of query processing has been ignored in all these studies, namely, aggregate queries. This is the problem we consider in this article. We provide a comprehensive study of query processing for a general model of imprecise data and a large class of aggregate queries.

We consider *probabilistic XML documents* and the unifying representation model of *p-documents* [Kimelfeld et al. 2008; Abiteboul et al. 2009]. A p-document can be viewed as a probabilistic process that randomly generates XML documents. Some nodes, namely distributional nodes, specify how to perform this random generation. We consider three kinds of distributional operators: *cie*, *mux*, *det*, respectively for *conjunction of independent events* (a node is selected if a conjunction of some probabilistic conditional events holds), *mutually exclusive* (at most one node selected from a set of a nodes), and *deterministic* (all nodes selected). This model, introduced by Kimelfeld et al. [2008], captures a large class of models for probabilistic trees that had been previously studied. For queries, we consider tree-pattern queries possibly with value joins and the restricted case of single-path queries. For aggregate functions, we consider the standard ones, namely, sum, count, min, max, countd (count distinct), and avg (average).

A p-document is a (possibly exponentially compact) representation of a probability space of (ordinary) documents, that is, a finite set of possible documents, each with a particular probability. In the absence of a grouping operation à la SQL (GROUP BY), the result of an aggregate query is a single value for each possible document. Therefore, an aggregate query over a p-document is a random variable and the result is a distribution, that is, the set of possible values, each with its probability. It is also interesting to consider summaries of the distribution of the result random variable (that is, possibly very large), in particular, its expected value and other probabilistic moments. When grouping is considered, a single value (again a random variable) is obtained for each match of the grouping part of the query. We investigate the computation of the distributions of random variables (in presence of grouping or not) and of their moments.

Our results highlight an (expectable) aspect of the different operators in p-documents: the use of *cie* (a much richer means of capturing complex situations) leads to a high complexity. For documents with *cie* nodes, we show the problems are hard (typically NP- or $FP^{\#P}$-complete). For count and sum, in the restricted setting of single-path queries, we show how to obtain moments in P. We also present Monte-Carlo methods that allow tractable approximations of probabilities and moments. On the other hand, with the milder forms of imprecision, namely *mux* and *det*, the complexity is lower. Computing the distribution for tree-pattern queries involving count, min and max is in P. The result distribution of sum may be exponentially large, but the computation is still in P in both *input and output*. On the other hand, computing avg or countd is $FP^{\#P}$-complete. On the positive side, we can compute expected values (and moments) for most aggregate tree-pattern queries in P. When we move to queries involving joins, the complexity of moment and distribution computation becomes $FP^{\#P}$-complete.

A main novelty of this work is that we also consider probabilistic XML documents involving continuous probability distributions, which capture a very frequent situation occurring in practice. We formally extend the probabilistic XML model by introducing leaves representing continuous value distributions. We explain how the techniques for the discrete case can be adapted to the continuous case and illustrate the approach on several classes of probability distributions.
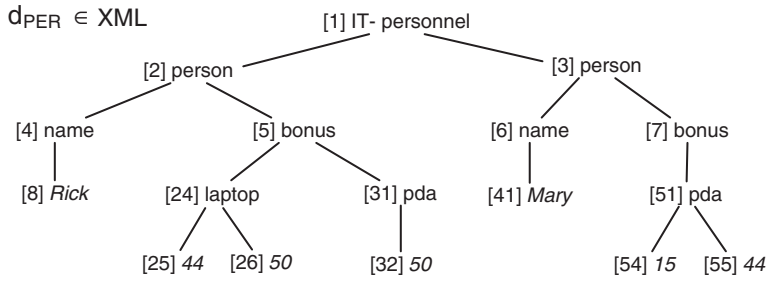
d_PER ∈ XML   [1] IT- personnel

[2] person   [3] person

[4] name   [5] bonus   [6] name   [7] bonus

[8] *Rick*   [24] laptop   [31] pda   [41] *Mary*   [51] pda

[25] *44* [26] *50*   [32] *50*   [54] *15* [55] *44*

Fig. 1.   Document d_{PER}: personnel in an IT department.

The article is organized as follows. In Section 2 we present preliminaries on deterministic data. In Section 3 we introduce the probabilistic data model, in terms both of discrete and continuous probability distributions. In Section 4 we define the three aggregation problems that are studied in the article and in Section 5 we discuss general principles we use to prove our results. The remaining sections are devoted to complexity analysis and algorithms for aggregation of probabilistic XML. Section 6 studies aggregation for the model with event variables. In Section 7, we investigate the particular case of *monoid aggregate functions* in the *mux-det* model, that is studied in full in Section 8. Since we show that many of the aggregation problems are intractable, we turn to approximation algorithms in Section 9. The related work is presented in Section 10. Because of space constraints, some of the proofs are regrouped in an appendix.

A preliminary version of some of this work appeared in Abiteboul et al. [2010]. New material includes in particular proofs, formalization of the results for the continuous case, examples, and insights into the semantics of aggregate queries over p-documents. Proofs of the most technical results are not included because of space constraints; they can be found in Kharlamov [2011].

2. DETERMINISTIC DATA AND QUERIES

We present here the data model and query languages we use.

Documents. We assume a countable set of *identifiers* $\mathcal{I}$ and a set of *labels* $\mathcal{L}$, such that $\mathcal{I} \cap \mathcal{L} = \varnothing$. The set of labels includes a set of data values (e.g., the rationals), on which the aggregate functions will be defined. A *document* is a pair $d = (t, \theta)$ of a finite, unordered[1] tree t, where each node has a unique identifier v and a label $\theta(v)$. We use the standard notions *child* and *parent*, *descendant* and *ancestor*, *root* and *leaf* in the usual way. To simplify the presentation, we assume that the leaves of documents are labeled with data values and the other nodes by nondata labels, that are called tags. The sets of *nodes* and *edges* of d are denoted, respectively, by $\mathcal{I}(d)$ and $\mathcal{E}(d)$, where $\mathcal{E}(d) \subseteq \mathcal{I}(d) \times \mathcal{I}(d)$. We denote the root of d by root(d).

Example 2.1.   Consider the document d_{PER} in Figure 1, where PER stands for *personnel*. 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_{PER} indicates Rick worked under two projects (laptop and pda) and got bonuses of 44 and 50 in the former project and 50 in the latter one.

------
[1]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 article.

Aggregate Functions. An *aggregate function* maps a finite bag of values (e.g., rationals) into some domain. In particular, we assume that any aggregate function is defined on the empty bag. In the article we study the common functions: sum, count, min, countd (count distinct), and avg (average) under the usual semantics. All results for min easily extend to max and top$K$.

More precisely,

—count and countd return the number of elements and the number of distinct elements in a bag, respectively; count($\{\!|\!\}$) = countd($\{\!|\!\}$) = 0.
—min returns the minimal element in the bag, and max returns the maximal element. min($\{\!|\!\}$) returns the special value $+\infty$, and max($\{\!|\!\}$) returns $-\infty$.
—sum and avg over bags of rational numbers compute their sum and average, respectively; by convention sum($\{\!|\!\}$) = avg($\{\!|\!\}$) = 0.

Aggregate functions can be naturally extended to work on documents d: the result $\alpha(d)$ is $\alpha(B)$ where B is the bag of the labels of all leaves in d. This makes the assumption that all leaves are of the type required by the aggregate function, for example, rational numbers for sum. Again to simplify, we ignore this issue here and assume they all have the proper type, say, rational numbers. It is straightforward to extend our models and results with a more refined treatment of typing.

As we will see some particular aggregate functions, the so-called *monoid* ones [Cohen et al. 2006], play a particular role in our investigation, because they can be handled by a divide-and-conquer strategy. Formally, a structure $(M, \oplus, \bot)$ is called an *Abelian monoid* if $\oplus$ is an associative and commutative binary operation with $\bot$ as identity element. If no confusion arises, we speak of the monoid M. An aggregate function is a *monoid* one if for some monoid M and any $a_1, \ldots, a_n \in M$.

$$\alpha(\{\!|a_1, \ldots, a_n|\!\}) = \alpha(\{\!|a_1|\!\}) \oplus \cdots \oplus \alpha(\{\!|a_n|\!\})$$

It turns out that sum, count, min, max, and top$K$ are monoid aggregate functions. For sum, min, max: $\alpha(\{\!|a|\!\}) = a$ and $\oplus$ is the corresponding obvious operation. For count: $\alpha(\{\!|a|\!\}) = 1$ and $\oplus$ is $+$. On the other hand, it is easy to check that neither avg nor countd are monoid aggregate functions.

Aggregate Queries over Documents. Finally, we introduce tree-pattern queries with joins, join-free queries, and single-path queries as special cases. We then extend them to aggregate queries.

We assume a countable set of variables *Var*. 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[2] 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, etc., in order to distinguish them from the nodes of documents, referred to as u, v, etc.

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 $E_/ \cup E_{/\!/}$ in a join-free query is a linear order, 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.

———————
[2]We denote the labeling function for queries as λ in order to distinguish it from the labeling function θ for documents.

$$Q_{\text{JoinBonus}}^{\text{avg}} \in \text{TPJ}^{\text{avg}} \qquad Q_{\text{RickBonus}}^{\text{sum}} \in \text{TP}^{\text{sum}} \qquad Q_{\text{Bonus}}^{\text{min}} \in \text{SP}^{\text{min}}$$

IT- personnel   IT- personnel   IT- personnel

person   person   person   bonus

Rick bonus   Mary bonus   Rick   bonus   [n] x

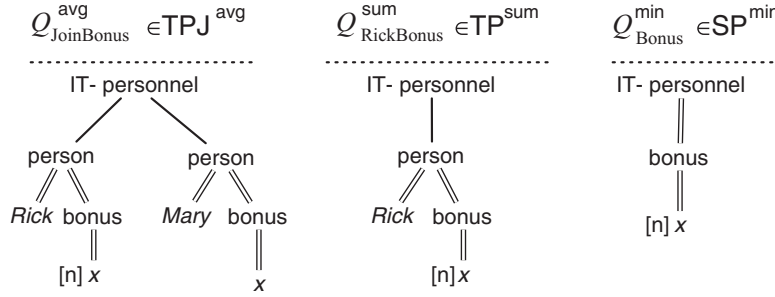[n] x   x   [n] x

Fig. 2.   Example aggregate queries over the document d_{PER} of Figure 1.

Example 2.2.   Figure 2 shows three example of tree-pattern queries for the document of Figure 1. Descendant edges are denoted with a double line. All three queries use a single variable, x, and have a single output node, marked with [n]. Disregard for now the line on the top with references to aggregate functions. The leftmost query, $Q_{\text{JoinBonus}}$, has a join variable; the query in the middle, $Q_{\text{RickBonus}}$, is a join-free query; the rightmost query, Q_{Bonus}, is single-path.

A *valuation* v maps query nodes to document nodes. A document *satisfies* a TPJ query if there exists a *satisfying* valuation, which maps query nodes to the document nodes in a way that is consistent with the edge types, the labeling, and the variable occurrences. That is, (1) the root of the query is mapped to the root of the document; (2) nodes connected by child/descendant edges are mapped to nodes that are children/descendants of each other; (3) query nodes with label a are mapped to document nodes with label a; and (4) two query nodes with the same variable are mapped to document nodes with the same label.

Slightly differently from other work, we define that applying a query $Q[\bar{n}]$ to a document d returns a set of tuples of nodes: $Q(d) := \{v(\bar{n}) \mid v \text{ satisfies } Q\}$. One obtains the more common semantics, according to which a query returns a bag of tuples of labels, by applying the labeling function of d to the tuples in $Q(d)$.

Example 2.3.   We now show the results of applying the queries of Figure 2 to the document of Figure 1. We denote u_i the node of d_{PER} with node identifier i. Query $Q_{\text{JoinBonus}}$ asks for bonuses of *Rick* that have the same value as one of *Mary*. Here, $Q_{\text{JoinBonus}}(d_{\text{PER}}) = \{u_{25}\}$. The second query retrieves all bonuses of *Rick*, that is, $\{u_{25}, u_{26}, u_{32}\}$. Finally, Q_{Bonus} retrieves all bonuses in the document and $Q_{\text{Bonus}}(d_{\text{PER}}) = \{u_{25}, u_{26}, u_{32}, u_{54}, u_{55}\}$.

An *aggregate* TPJ query has the form $Q[\alpha(n)]$, where Q is a tree pattern, n is a leaf node of Q, and α is an aggregate function. We evaluate such $Q[\alpha(n)]$ in three steps.

(1) First, we evaluate the nonaggregate component $Q' := Q[n]$ of $Q[\alpha(n)]$ over d in the following way:

$$Q'(d) = \{v(n) \mid v \text{ satisfies } Q \text{ and } v(n) \text{ is a leaf of } d\}.$$

Note that evaluation of nonaggregate components of $Q[\alpha(n)]$ is defined differently from evaluation of TPJ queries: we have an extra requirement that $v(n)$ is a leaf of d. We do it because we want to aggregate *values* stored in the leaves of documents but not tags of the internal nodes.

(2) We then compute the *bag B* of labels of $Q'(d)$, that is

$$B := \{\!\{ \theta(n) \mid n \in Q'(d) \}\!\}.$$

(3) Finally we apply α to B and

$$Q[\alpha(n)](d) = \alpha(B).$$

Identifying the aggregate query with its pattern, we denote the value resulting from evaluating Q over d as $Q(d)$.

If $Q[n]$ is a nonaggregate query and α an aggregate function, we use the shorthand $Q^\alpha[n]$ to denote the aggregate query $Q[\alpha(n)]$. Similarly, we denote the set of aggregate queries obtained from queries in TPJ, TP, SP and some function α, as TPJ^α, TP^α, SP^α, respectively.

The syntax and semantics just given can be generalized in a straightforward fashion to aggregate queries with SQL-like GROUP BY. Such queries are written $Q[\bar{n}, \alpha(n)]$ and return an aggregate value for every binding of $\bar{n}$ to a tuple of document nodes. Since we can reduce the evaluation of such queries to the evaluation of several simpler queries of the kind defined before, while increasing the data complexity by no more than a polynomial factor, we restrict ourselves to that simpler case.

Example 2.4. Continuing with Example 2.2, we compute here the results of the aggregate queries of Figure 2 over d_{PER}. The aggregate function used is given on the line above each query. Let us start with $Q^{\mathsf{avg}}_{\mathrm{JoinBonus}}$: since the corresponding nonaggregate query has a single result, $Q^{\mathsf{avg}}_{\mathrm{JoinBonus}}(d_{\mathrm{PER}}) = \theta(Q_{\mathrm{JoinBonus}}(d_{\mathrm{PER}})) = 44$. Consider now $Q^{\mathsf{sum}}_{\mathrm{RickBonus}}$. We have: $Q^{\mathsf{sum}}_{\mathrm{RickBonus}}(d_{\mathrm{PER}}) = \mathsf{sum}(\{\!\!\{44, 50, 50\}\!\!\}) = 144$. Finally, $Q^{\mathsf{min}}_{\mathrm{Bonus}}(d_{\mathrm{PER}}) = \mathsf{min}(\{\!\!\{44, 50, 50, 15, 44\}\!\!\}) = 15$.

3. PROBABILISTIC DATA

We now introduce our uncertainty model. We first introduce the discrete probabilistic XML model from Kimelfeld et al. [2008] and Abiteboul et al. [2009] and then extend it to support continuous probability distributions.

3.1. Discrete Probabilistic Data

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

p-Documents: Syntax. Following Abiteboul et al. [2009], we now introduce a very general syntax for compactly representing px-spaces, called *p-documents*. p-documents are similar to documents, with the difference that they have 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.

More precisely, we assume given a set $\mathcal{X}$ of Boolean random variables with some specified probability distribution Δ over them. A *p-document*, denoted by $\widehat{\mathcal{P}}$, is an unranked, unordered, labeled tree. Each node has a unique identifier v and a label $\mu(v)$ in $\mathcal{L} \cup \{cie(E)\}_E \cup \{mux(\mathsf{Pr})\}_{\mathsf{Pr}} \cup \{det\}$ where $\mathcal{L}$ are labels of *ordinary* nodes, and the others are labels of *distributional* nodes. We consider three kinds of the latter labels: $cie(E)$ (for conjunction of independent events), $mux(\mathsf{Pr})$ (for mutually exclusive), and det (for deterministic). We will refer to distributional nodes labeled with these labels, respectively, as *cie*, *mux*, and *det* nodes. If a node v is labeled with $cie(E)$, then E is a function that assigns to each child of v a conjunction $e_1 \wedge \cdots \wedge e_k$ of literals (x or $\neg x$, for $x \in \mathcal{X}$). If v is labeled with $mux(\mathsf{Pr})$, then Pr assigns to each child of v a probability with the sum across all children less than or equal to 1.
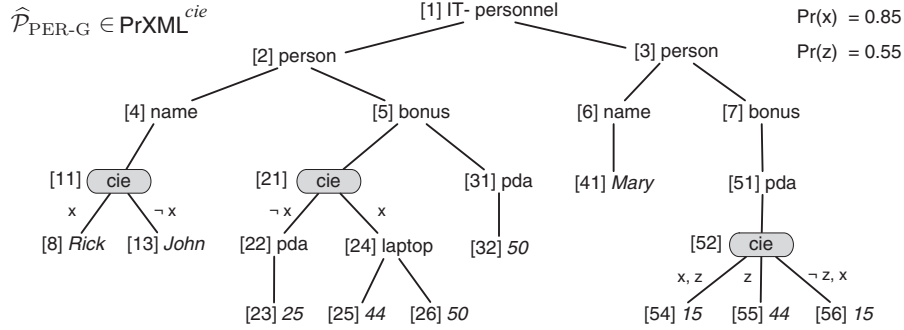
Fig. 3.   PrXML^{cie} p-document: IT department.
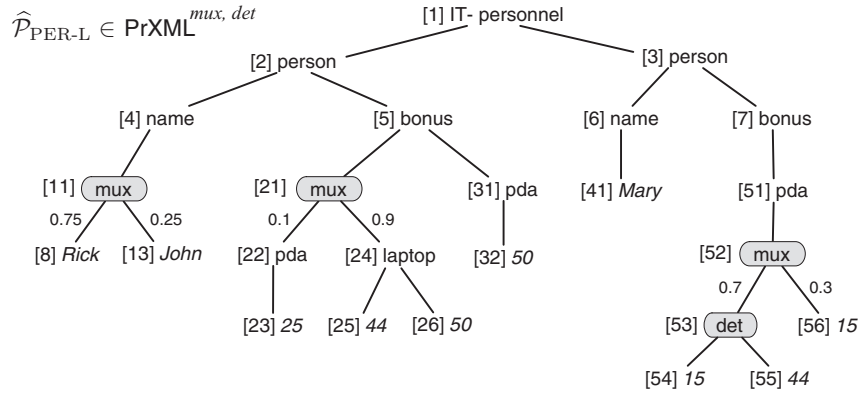
Fig. 4.   PrXML^{mux,det} p-document: IT department.

Example 3.1. Two p-documents are shown in Figures 3 and 4. The former, $\widehat{\mathcal{P}}_{\text{PER-G}}$ has only *cie* 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 $E(n_{24}) = x$. The p-document from Figure 4, $\widehat{\mathcal{P}}_{\text{PER-L}}$ has only *mux* and *det* distributional nodes. Node n_{52} has label *mux*(Pr) and two children n_{53} and n_{56}, where $\mathsf{Pr}(n_{53}) = 0.7$ and $\mathsf{Pr}(n_{56}) = 0.3$. The letters G and L in the name of these p-documents stand for *global* and *local*, which describes the probabilistic dependencies captured by PrXML^{cie} and PrXML^{mux,det}.

We denote *classes of p-documents* by PrXML with a superscript denoting the types of distributional nodes that are allowed for the documents in the class. For instance, PrXML^{mux,det} is the class of p-documents with only *mux* and *det* distributional nodes, like $\widehat{\mathcal{P}}_{\text{PER-L}}$.

In the sequel, we will sometimes refer to p-documents from PrXML^{mux,det,cie} as *discrete p-documents*, to distinguish them from the continuous p-documents to be defined further.

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 obtainable from $\widehat{\mathcal{P}}$ by a randomized three-step process.

(1) We choose a valuation v of the variables in $\mathcal{X}$. The probability of the choice, according to the distribution Δ, is

$$p_v = \prod_{\substack{x \text{ in } \mathcal{X} \\ v(x)=\text{true}}} \Delta(x) \;\cdot\; \prod_{\substack{x \text{ in } \mathcal{X} \\ v(x)=\text{false}}} (1 - \Delta(x)).$$

(2) For each *cie* node labeled *cie*(E), we delete its children v where $v(E(v))$ is false, and their descendants. Then, independently for each *mux* node v labeled *mux*(Pr), we choose one of its children v' (with probability Pr) or none at all (with probability $1 - \sum_v \text{Pr}(v)$) and delete the other children and their descendants. We do not delete any of the children of *det* nodes.[3]

(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'.

The result of this third step is a random document $\mathcal{P}$. The probability $\text{Pr}(\mathcal{P})$ is defined as the product of p_v with all probabilities of choices we made in the second step for the *mux* nodes.

Example 3.2. One can obtain the document d_{PER} in Figure 1 by applying the randomized process to the p-document in Figure 4. Then the probability of d is $\text{Pr}(d) = 0.75 \times 0.9 \times 0.7 = 0.4725$. One can also obtain d from the p-document in Figure 3, by assigning $\{x/\text{true}, z/\text{true}\}$. In this case the probability of d is $\text{Pr}(d) = 0.85 \times 0.55 = 0.4675$.

As shown in Abiteboul et al. [2009], both PrXML^{mux,det} and PrXML^{cie} have full expressive power with respect to discrete px-spaces: every discrete probability space over documents can be obtained as the semantics of a p-document of PrXML^{mux,det} or of PrXML^{cie}. Furthermore, PrXML^{cie} is exponentially more concise than PrXML^{mux,det}: all PrXML^{mux,det} p-documents can be transformed into equivalent PrXML^{cie} in polynomial time, but some PrXML^{cie} have only exponential-size equivalent PrXML^{mux,det} p-documents. This exponential concision comes at a cost: it was shown in Kimelfeld et al. [2008, 2009] that the data complexity of answering TP-queries is FP$^{\#P}$-complete for PrXML^{cie}, whereas it is in P for PrXML^{mux,det}; see details on query answering in Section 4.1.

In our analysis, we only consider distributional nodes of the types *cie*, *mux*, and *det*. In Abiteboul et al. [2009] two more types of distributional nodes (*ind* and *exp*) are considered. As shown there, the first kind can be captured by *mux* and *det*, while the second is a generalization of *mux* and *det*; most results for PrXML^{mux,det} can be extended to PrXML^{exp}.

3.2. Continuous Probabilistic Data

We generalize p-documents to documents whose leaves are labeled with (representations of) probability distributions over the reals, instead of single values. We give semantics to such documents in terms of continuous distributions over documents with real numbers on their leaves.

In the discrete case, a p-document defines a finite set of trees and probabilities assigned to them. In the continuous case, a p-document defines a continuous px-space consisting of an uncountably infinite set of trees $\mathcal{D}$ with a σ-algebra $\mathcal{A}$ and a probability measure Pr on $\mathcal{A}$. Thus, a continuous px-space is a probability space $(\mathcal{D}, \mathcal{A}, \text{Pr})$ over an infinite set of trees. Not that in this setting, it is possible that the probability of any

---

[3]It may seem that *det* nodes are redundant, but they actually increase expressive power when used together with *mux* and other types of distributional nodes [Abiteboul et al. 2009].
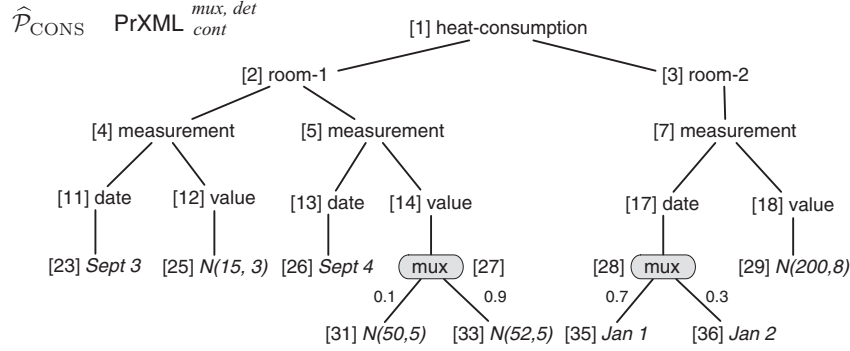
Fig. 5. PrXML$_{cont}^{mux,det}$ p-document: monitoring.

single document $\mathcal{P} \in \mathcal{D}$ is zero. We refer to a textbook on measure and probability theory such as Ash and Doléans-Dade [2000] for the definitions of the concepts used in this section.

Continuous p-documents: Syntax. To support continuous distributions on leaves, we extend the syntax of p-documents by an additional type of distributional nodes, the *cont* nodes. A *cont* node has the form *cont*(D), where D is a representation of a probability distribution over the real numbers. In contrast to the distribution nodes introduced earlier, a *cont* node can only appear as a leaf. Given a *cont* leaf l, we denote with f_l the *probability density function* (*pdf* for short) of the distribution attached to l.

We denote classes of p-documents where *cont* nodes can appear with a *cont* subscript, for example, PrXML$_{cont}^{mux,det}$ or PrXML$_{cont}^{cie}$.

Example 3.3. Consider the PrXML$_{cont}^{mux,det}$ p-document in Figure 5. The document collects results of heat consumption monitoring in two rooms. Since the measurements are imprecise (with an imprecision that grows the higher the heat consumption is), they are given as normal distributions $N(\mu, \sigma^2)$ (i.e., Gaussians), centered around the mean μ and with variance σ^2. The actual temperature is unknown, but this distribution gives its probability distribution, for instance, as specified by the sensor manufacturer. Additionally, there is a discrete source of uncertainty because records are ambiguous whether the measurement in the second room was done on January 1st or 2nd.

Any finitely representable distribution can appear in a *cont* node. As an example, we consider in the following piecewise polynomial distributions. A function $f \colon \mathbb{R} \to \mathbb{R}$ is piecewise polynomial if there are points $-\infty = x_0 < x_1 < \cdots < x_m = \infty$ such that for each open interval $I_i :=]x_{i-1}, x_i[$, $1 \leqslant i \leqslant m$, the restriction $f_{|I_i}$ of f to I_i is a polynomial. (The points $x_1, \ldots, x_{n-1}$ are the partition points and the intervals $I_1, \ldots, I_m$ are the partition intervals of f.) Every piecewise polynomial function $f \geqslant 1$ with $\int_{-\infty}^{\infty} f = 1$ is the density function of a probability. Clearly, in this case $f_{|I_1}$ and $f_{|I_m}$ are identical to 0. Note that distributions defined by piecewise polynomial densities are a generalization of uniform distributions. Piecewise polynomials are an example of a class of functions stable under convex sum, (classical) convolution, product, and integration. We shall use this stability property later on to compute the distribution of aggregate query answers. We shall refer to the class of piecewise polynomials of degree bounded by an integer K as PP(K); PP(0) is thus the class of piecewise uniform distributions.

Continuous p-documents: Semantics. A continuous p-document $\widehat{\mathcal{P}}$ is interpreted as a continuous px-space $[\![\widehat{\mathcal{P}}]\!] = (\mathcal{D}, \mathcal{A}, \mathsf{Pr})$, where $\mathcal{D}$ is an infinite set of documents, $\mathcal{A}$ is a

σ-algebra over $\mathcal{D}$, and Pr is a probability measure over $\mathcal{A}$. We construct this triple by a sequence of steps.

We begin by nondeterministically expanding the distributional nodes of type *cie*, *mux*, and *det* in $\widehat{\mathcal{P}}$ in the way described in Section 3.1. This results in a finite collection of p-documents $\widehat{\mathcal{P}}_1, \ldots, \widehat{\mathcal{P}}_n \in \mathsf{PrXML}_{cont}$, called the *skeletons* of $\widehat{\mathcal{P}}$, each with a probability p_i.

Given the skeletons, we will define: (1) the elements of the final continuous space $\mathcal{D}$; (2) the σ-algebra $\mathcal{A}$; (3) the probability measure Pr. We will do this by first defining for each $\widehat{\mathcal{P}}_i$ a document set $\mathcal{D}_i$, a σ-algebra $\mathcal{A}_i$ on $\mathcal{D}_i$, and a probability measure Pr_i on $\mathcal{A}_i$, from which we construct the final $\mathcal{D}$, $\mathcal{A}$, and Pr. The intuition behind our approach is that a skeleton with k *cont* leaves labeled with distributions resembles a k-dimensional Euclidean space $\mathbb{R}^k$ and that we can transfer the σ-algebra and the probability measure resulting from distributions from $\mathbb{R}^k$ to a space of documents with real numbers on the leaves.

We now consider a fixed skeleton $\widehat{\mathcal{P}}_i$. We define $\mathcal{D}_i$ as the set of trees obtained by substituting the *cont* nodes of $\widehat{\mathcal{P}}_i$ with real numbers in all possible ways. From the $\mathcal{D}_i$'s we construct the final space as $\mathcal{D} := \bigcup_{i=1}^{n} \mathcal{D}_i$.

Next we define on each $\mathcal{D}_i$ a σ-algebra $\mathcal{A}_i$ and a probability measure Pr_i, from which we construct in a second step $\mathcal{A}$ and Pr. Let $(l_1, \ldots, l_k)$ be the *cont* leaves of $\widehat{\mathcal{P}}_i$, where l_j is labeled with the distribution D_j. (Note that this includes the case where $k = 0$.) There is a natural bijection between the space $\mathbb{R}^k$ of k-tuples of real numbers and the space $\mathcal{D}_i$, consisting of all documents resulting from $\widehat{\mathcal{P}}_i$ by labeling the leaf nodes l_j with real numbers. In fact, let $\mathcal{P}_i$ be the function that maps the tuple $(x_1, \ldots, x_k) \in \mathbb{R}^k$ to the document obtained by labeling each l_j with x_j. Clearly, $\mathcal{P}_i : \mathbb{R}^k \to \mathcal{D}_i$ is bijective. By means of $\mathcal{P}_i$, we can now transfer the σ-algebra of Borel sets in $\mathbb{R}^k$ to $\mathcal{D}_i$. We thus define that $\mathcal{A}_i$ consists of those sets $\mathcal{D}' \subseteq \mathcal{D}_i$ such that the preimage $\mathcal{P}^{-1}(\mathcal{D}')$ is a Borel set in $\mathbb{R}^k$. Since the Borel sets constitute a σ-algebra, $\mathcal{A}_i$ is also a σ-algebra. From the $\mathcal{A}_i$'s we construct the final algebra $\mathcal{A}$ as the collection of sets $\mathcal{A} := \{\mathcal{D}'_1 \cup \cdots \cup \mathcal{D}'_n \mid \mathcal{D}'_i \in \mathcal{A}_i\}$, that is, a set in $\mathcal{A}$ is the (disjoint) union of sets taken from the $\mathcal{A}_i$.

To define Pr_i, let $D_1, \ldots, D_k$ be the k probability distributions represented in the *cont* nodes of $\widehat{\mathcal{P}}_i$. We then define the measure $\widehat{\mathsf{Pr}}_i$ over $\mathbb{R}^k$ as the product measure [Ash and Doléans-Dade 2000] induced by the D_j, that is, the unique measure such that $\widehat{\mathsf{Pr}}_i(X_1 \times \cdots \times X_k) = D_1(X_1) \times \cdots \times D_k(X_k)$ for all $X_j \subseteq \mathbb{R}$. The measure $\widehat{\mathsf{Pr}}_i$ is a probability measure, since all D_j's induce a probability measure. We note that with respect to the product measure, the values in different dimensions of $\mathbb{R}^k$ are independent. Using again the inverse of the bijection $\mathcal{P}_j$ introduced earlier, we translate $\widehat{\mathsf{Pr}}_i$ into a probability measure Pr_i over $\mathcal{A}_i$ by defining $\mathsf{Pr}_i(\mathcal{D}'_i) = \widehat{\mathsf{Pr}}_i(\mathcal{P}_i^{-1}(\mathcal{D}'_i))$ for all $\mathcal{D}'_i \in \mathcal{A}_i$. Since $\mathcal{A}_i$ consists exactly of translations of Borel sets over the real numbers under $\mathcal{P}_i$, this defines a probability measure over $\mathcal{A}_i$. Thus, for each skeleton $\widehat{\mathcal{P}}_i$ we have defined a probability space $[\![\widehat{\mathcal{P}}_i]\!] = (\mathcal{D}_i, \mathcal{A}_i, \mathsf{Pr}_i)$.

We now combine the spaces of the skeletons and define the probability space represented by $\widehat{\mathcal{P}}$ as $[\![\widehat{\mathcal{P}}]\!] = (\mathcal{D}, \mathcal{A}, \mathsf{Pr})$, where $\mathcal{D}$, as defined previously, is the disjoint union of the $\mathcal{D}_i$'s, $\mathcal{A}$, as already defined, consists of unions of sets from the $\mathcal{A}_i$'s, and Pr is the convex sum of the Pr_i's, that is,

$$\mathsf{Pr}(\mathcal{D}') = \sum_{i=1}^{n} p_i \cdot \mathsf{Pr}_i(\mathcal{D}' \cap \mathcal{D}_i)$$

for every $\mathcal{D}' \in \mathcal{A}$. Intuitively, this means that a set $\mathcal{D}' \in \mathcal{A}$ is decomposed into subsets that are in the $\mathcal{A}_i$'s, then the Pr_i's are applied to each subset, and the results are combined as a weighted sum where the weight of the i-th component is the probability

of the i-th skeleton $\widehat{\mathcal{P}}_i$. Since the Pr_i's are probability measures and $p_1 + \cdots + p_n = 1$, Pr is a probability measure.

We summarize the construction by saying that $[\![\widehat{\mathcal{P}}]\!]$ is a *convex sum* of the $[\![\widehat{\mathcal{P}}_i]\!]$'s and write

$$[\![\widehat{\mathcal{P}}]\!] = p_1 \cdot [\![\widehat{\mathcal{P}}_1]\!] + \cdots + p_n \cdot [\![\widehat{\mathcal{P}}_n]\!]. \tag{1}$$

A p-document of $\mathsf{PrXML}_{cont}^{cie,mux,det}$ can have (leaf) nodes annotated with discrete values as well as with continuous distributions. To simplify the presentation, however, we will assume that all nodes of a $\mathsf{PrXML}_{cont}^{cie,mux,det}$ p-document returned by a nonaggregate query or aggregated by an aggregate query (see the following section for definitions) are *cont* nodes. Adding support for aggregating both discrete and continuous nodes in the same query boils down to representing discrete values with *Dirac distributions* and manipulating these Dirac distributions (through convolutions, convex sums, etc.) in the same way as continuous probability distributions, using the tools of distribution theory [Friedlander and Joshi 1999]. All our complexity upper bounds can be extended this way to the case of documents combining continuous and discrete probability distributions of values.

4. AGGREGATION OF PROBABILISTIC DATA

We briefly review the semantics of nonaggregate queries over px-spaces. We define the result of an aggregate query over a p-document as a distribution over the answer values. In the continuous case we discuss the mathematical relation between the answer distribution and the distributions on the *cont*-nodes of the input document. Finally, we define the computational problems that we want to study, both for discrete and continuous data.

4.1. Nonaggregate Queries over Probabilistic Data

When studying nonaggregate queries over probabilistic documents, researchers have mainly studied Boolean queries because answering queries with output variables can be reduced to this case with polynomial-time data complexity.

Consider a Boolean TPJ query Q. Clearly, when evaluated over a deterministic document d, either d satisfies Q or does not, thus, the result is either "true" or "false". Now, let $\mathcal{S} = (\mathcal{D}, \mathcal{A}, \mathsf{Pr})$ be a px-space of documents. Then Q is satisfied by some documents $d \in \mathcal{D}$, but not by others. Consequently, one defines the result of evaluating Q over $\mathcal{S}$ as the number $Q(\mathcal{S}) = \mathsf{Pr}(\{d \in \mathcal{S} \mid d \models Q\})$, the *probability* that a document satisfies Q.[4] So far, researchers have investigated query evaluation over spaces represented by discrete p-documents $\widehat{\mathcal{P}}$. Then $[\![\widehat{\mathcal{P}}]\!]$ contains finitely many documents $d_1, \ldots, d_n$, each with a probability $\mathsf{Pr}(d_i)$, and evaluating Q over $\widehat{\mathcal{P}}$ results in $Q(\widehat{\mathcal{P}}) = \sum_{\substack{d \in \widehat{\mathcal{P}} \\ d \models Q}} \mathsf{Pr}(d)$. Table I summarizes the data complexity of computing $Q(\widehat{\mathcal{P}})$ over discrete p-documents. Results for SP and TP queries come from Kimelfeld et al. [2009] while $\mathsf{FP}^{\#\mathsf{P}}$-completeness for TPJ queries is shown in Section 5.

Over a discrete space $\mathcal{S}$, where every document has a nonnegative probability, a document satisfying Q contributes a nonnegative fragment to the result $Q(\mathcal{S})$. Over a continuous space it is possible that $Q(\mathcal{S}) > 0$, but $\mathsf{Pr}(\{d\}) = 0$ for every single document that satisfies Q.

---

[4]Technically, the definition assumes that the set of documents satisfying Q is an element of $\mathcal{A}$, which trivially holds in our setting.

Table I. Data Complexity of Nonaggregate Query Evaluation over
Discrete p-Documents

	Non-aggregate query language	
	SP and TP	TPJ
cie	$FP^{\#P}$-complete [Kimelfeld et al. 2009]	$FP^{\#P}$-complete
mux, *det*	in P [Kimelfeld et al. 2009]	$FP^{\#P}$-complete

4.2. Aggregate Queries over Probabilistic Data: Semantics

For the sake of simplicity, we assume that aggregate functions take real numbers as values. Let $\mathcal{S} = (\mathcal{D}, \mathcal{A}, \mathsf{Pr})$ be an arbitrary px-space and Q^α be an aggregate query.

Aggregate Queries over Discrete Documents. If $\mathcal{D}$ is finite, $\mathcal{D} = \{d_1, \ldots, d_n\}$, then there are finitely many real numbers $c_1, \ldots, c_m$, $m \leqslant n$, that occur as values of Q^α. The probability that c_j is the value of Q^α is $\sum\{\mathsf{Pr}(d_i) \mid Q^\alpha(d_i) = c_j\}$, the sum of the probabilities of the documents over which Q^α returns the aggregate value c_j.

Accordingly, we define the *discrete distribution* $Q^\alpha(\mathcal{S})$ as the function $\mathbb{R} \to [0, 1]$, satisfying

$$Q^\alpha(\mathcal{S})(c) = \sum \big\{\mathsf{Pr}(d) \mid d \in \mathcal{D},\ Q^\alpha(d) = c\big\}.$$

Clearly, $Q^\alpha(\mathcal{S})(c) = 0$ iff $c \notin \{c_1, \ldots, c_m\}$. We call the set of values of Q^α over $\mathcal{D}$ that have nonzero probability the *carrier* of the distribution $Q^\alpha(\mathcal{S})$.

Example 4.1. Evaluation of the query $Q^{\min}_{\text{Bonus}}$ from Figure 2 over the *cie* p-document $\widehat{\mathcal{P}}_{\text{PER-G}}$ in Figure 3 gives the distribution $\{(15, 0.85), (25, 0.15)\}$, since in two out of four worlds corresponding to the assignments $\{x/\mathsf{false}, z/\mathsf{false}\}$ and $\{x/\mathsf{false}, z/\mathsf{true}\}$, with probabilities 0.0675 and 0.0825, respectively, the minimum bonus is 25 and in the remaining two worlds the minimum is 15. Evaluation of the query over the *mux-det* p-document $\widehat{\mathcal{P}}_{\text{PER-L}}$ in Figure 4 gives the distribution $\{(15, 1)\}$, since in every world of the p-document Mary receives a bonus of 15 which is the smallest value across all bonuses occurring in the p-document.

Aggregate Queries over Continuous Documents. In the general case, Q^α maps the elements of the probability space $\mathcal{S}$ to real numbers, and therefore can be seen as a function $Q^\alpha \colon \mathcal{D} \to \mathbb{R}$. An aggregate query Q^α is a *total* function on $\mathcal{D}$ because we require that aggregate functions are also defined for the empty bag (see Section 2) and thus Q^α produces a value for every $d \in \mathcal{D}$ even if the underlying nonaggregate query is not satisfied by d. A real-valued function defined on a probability space is a *random variable*.[5]

The random variable Q^α induces a probability measure Pr' on the reals, defined for measurable sets $A \subseteq \mathbb{R}$ and satisfying $\mathsf{Pr}'(A) = \mathsf{Pr}(\{d \in \mathcal{D} \mid Q^\alpha(d) \in A\})$. Less formally, Pr' answers the question, "what is the probability that the value of Q^α is an element of A?"

The probability Pr' can be captured by two functions, the *cumulative* distribution $F \colon \mathbb{R} \to \mathbb{R}$ where $F(x) = \mathsf{Pr}(Q^\alpha(d) \leqslant x) = \mathsf{Pr}'(] - \infty, x])$, and the *distribution function*

$$Q^\alpha(\mathcal{S})(x) = \frac{d}{dx} \mathsf{Pr}(Q^\alpha(d) \leqslant x),$$

which is the first derivative of F and is also sometimes called the probability density function of Pr'. (We study only *continuous* probabilities, where F is differentiable.) We define the distribution function $Q^\alpha(\mathcal{S})$ as the *result* of evaluating Q^α over $\mathcal{S}$.

---

[5]This is not fully correct, since technically, the function has to be measurable, a condition that is always fulfilled in our setting.

If $\mathcal{S}$ is represented by the p-document $\widehat{\mathcal{P}}$, we write $Q^\alpha(\widehat{\mathcal{P}})$ instead of $Q^\alpha(\llbracket\widehat{\mathcal{P}}\rrbracket)$. Moreover, when we refer to Q^α as a random variable and want to stress that the arguments of Q^α are random documents obtained from $\widehat{\mathcal{P}}$, we write $Q^\alpha(\mathcal{P})$.

Following the notation $\alpha(d)$ introduced to denote the aggregation of all leaves of a document, we denote $\alpha(\mathcal{S})$ or $\alpha(\widehat{\mathcal{P}})$ the aggregation of all leaves of a px-space or of a p-document, that is, $\alpha(\mathcal{S}) := Q_0^\alpha(\mathcal{S})$ and $\alpha(\widehat{\mathcal{P}}) := Q_0^\alpha(\widehat{\mathcal{P}})$ for Q_0 the SP query "//*" that returns all nodes of a document (remember that, in the semantics of aggregate queries, only *leaf* values are aggregated).

4.3. Continuous Aggregation

We now want to study the mathematical relationship between the distribution $Q^\alpha(\widehat{\mathcal{P}})$ and the leaf distributions of $\widehat{\mathcal{P}}$.

We consider first aggregation over skeletons, that is, p-documents whose only distributional nodes are of type *cont*. We assume that every leaf is labeled with a continuous distribution, to avoid complications caused by the combination of continuous and discrete distributions. Then we generalize the results to aggregate queries over skeletons, and finally to queries over arbitrary p-documents.

Since we admit arbitrary continuous functions as distributions, provided they are nonnegative and have an integral of 1, we restrict the aggregate functions to those common in data management. We first note that aggregation of count does not depend on labels and therefore is the same in the discrete and the continuous case. We will see later on that, in the continuous case, aggregation with countd is essentially the same as with count. Therefore, we limit ourselves to aggregation with $\alpha \in \{\mathsf{sum}, \mathsf{min}, \mathsf{max}, \mathsf{avg}\}$.

Aggregation over Skeletons. Let $\widehat{\mathcal{P}}$ be a skeleton whose leaves $l_1, \ldots, l_n$ are labeled by the continuous distributions $f_1, \ldots, f_n$, and let $\llbracket\widehat{\mathcal{P}}\rrbracket = (\mathcal{D}, \mathcal{A}, \mathsf{Pr})$ be the px-space it generates. Let $\mathcal{P}\colon \mathbb{R}^n \to \mathcal{D}$ be the labeling function where $\mathcal{P}(x_1, \ldots, x_n)$ is obtained from $\widehat{\mathcal{P}}$ by labeling the leaf l_i with the number x_i: this defines a bijection. Both the algebra structure $\mathcal{A}$ and the probability Pr on $\mathcal{D}$ are exactly the images of the Borel algebra and of the probability $\widehat{\mathsf{Pr}}$ on $\mathbb{R}^n$, where $\widehat{\mathsf{Pr}}$ is the product of the probability measures induced by the distributions f_i. This means, in more concrete terms, $\widehat{\mathsf{Pr}}$ is the measure with density function $f(x_1, \ldots, x_n) = f_1(x_1) \times \cdots \times f_n(x_n)$.

The distribution of the aggregate values of α over $\mathcal{P}(\bar{x}) \in \mathcal{D}$ is therefore the same as the one of applying α to all bags resulting from tuples $\bar{x} \in \mathbb{R}^n$, taking into account the density $f(\bar{x})$.

We recall that the cumulative distribution of a distribution function g is defined as $G(x) = \int_{-\infty}^{x} g(y)\,dy$ and thus $g = G'$. Moreover, the convolution of two distributions g, h is defined as $(g * h)(x) = \int_{-\infty}^{\infty} g(y) \cdot h(x - y)\,dy$.

PROPOSITION 4.2. *Let X, Y be independent real-valued random variables with distribution functions g, h and cumulative distributions G, H. Then:*

(1) *The density function of $X + Y$ is $g * h$.*
(2) *The cumulative distribution function of $\mathsf{max}(X, Y)$ is $G \cdot H$.*
(3) *The cumulative distribution function of $\mathsf{min}(X, Y)$ is $G + H - G \cdot H$.*

PROOF. Claim 1 is a classical result in probability theory [Ash and Doléans-Dade 2000]. To see Claim 2, note that $\mathsf{max}(X, Y) \leqslant x$ if and only if $X \leqslant x$ and $Y \leqslant x$. The probability for the first condition is $G(x)$, for the second $H(x)$. Since X, Y are independent, the events $X \leqslant x$ and $Y \leqslant x$ are independent. The probability for both of them being true is therefore the product of the two probabilities. To see Claim 3, note that $\mathsf{min}(X, Y) > x$ if and only if $X > x$ and $Y > x$. Therefore this condition has the

probability $p = (1 - G(x))(1 - H(x))$ and the complementary condition $\min(X, Y) \leqslant x$ has probability $1 - p = G(x) + H(x) - G(x) \cdot H(x)$. □

As seen in the proposition, classical convolution expresses the distribution of sum. In analogy, we define max- and min-convolutions of distribution functions g, h as the derivatives of the cumulative distributions of max and min appearing before: $g *_{\max} h = g \cdot H + G \cdot h$ and $g *_{\min} h = g + h - g \cdot H - G \cdot h$, where G, H are the cumulative distributions of g, h, respectively. Moreover, we will denote classical convolution also as $g *_{\text{sum}} h$. Since all $\alpha \in \{\text{sum}, \text{max}, \text{min}\}$ are based on associative and commutative binary operations, also the corresponding convolution operations "$*_\alpha$" are associative and commutative.

PROPOSITION 4.3. *Let $\widehat{\mathcal{P}}$ be a skeleton with distributions $f_1, \ldots, f_n$ on the leaves.*

(1) *If α is one of* sum, min, max, *then the distribution $\alpha(\widehat{\mathcal{P}})$ is the n-fold α-convolution of the f_i's, that is $\alpha(\widehat{\mathcal{P}}) = f_1 *_\alpha \cdots *_\alpha f_n$.*
(2) *For* avg, *the distribution is* $\text{avg}(\widehat{\mathcal{P}})(x) = n \cdot f(n \cdot x)$, *where f is the n-fold* sum-*convolution of the f_i's.*
(3) *For* countd, *the probability that all leaves have different labels is 1, that is,* $\Pr(\text{countd}(P) = n) = 1$.

PROOF. As discussed earlier, instead of trees with real numbers as leaf labels, we can immediately apply α to tuples in $\mathbb{R}^n$. Then Claim 1 follows from Proposition 4.2 and the definition of α-convolutions. For avg, the number of values aggregated is always n. Therefore, $\text{avg}(\mathcal{P}(\bar{x})) = \text{sum}(\mathcal{P}(\bar{x}))/n$. Then Claim 2 follows, since for a random variable X with distribution $g(x)$, the random variable X/n has the distribution $n \cdot g(n \cdot x)$. A tuple $\bar{x}$ has less than n distinct entries if $\bar{x}$ is an element of a hyperplane satisfying an equation "$x_i = x_j$" over $\mathbb{R}^n$, for some $1 \leqslant i < j \leqslant n$. The Borel measure of a hyperplane is 0. Since $\widehat{\Pr}$ is defined by a continuous distribution, $\widehat{\Pr}(x_i = x_j) = 0$. Then Claim 3 follows, since there are only finitely many such hyperplanes. □

For the same reason that countd is not a useful aggregate function for continuous p-documents, joins of leaves are not of much interest: the probability that two continuous leaves are equal is zero. Future work could study more adapted query languages, for example, tree-pattern queries with inequalities.

Aggregate Queries over Skeletons. In addition to the skeleton $\widehat{\mathcal{P}}$, we now consider also a TPJ aggregate query Q^α. We drop the assumption that all leaves carry distributions, but assume that if Q retrieves a node v over some $Q(\bar{x})$, then v is a *cont*-node in $\widehat{\mathcal{P}}$.

While before, when evaluating α directly over the documents $\mathcal{P}(\bar{x})$, we aggregated the labels of *all* leaves, we now only aggregate the labels of leaf nodes returned by Q. A leaf v is returned by Q if there is a satisfying valuation ν from Q to $\mathcal{P}(\bar{x})$ such that $\nu(n) = v$, where n is the output node of Q. Since all documents $\mathcal{P}(\bar{x})$ have the same nodes and the same tree structure as $\widehat{\mathcal{P}}$ we can view valuations also as mappings from Q to $\widehat{\mathcal{P}}$. Clearly, we need not consider leaf nodes that are only returned with probability 0. Intuitively, such nodes are retrieved by valuations that satisfy Q because a leaf label of $\mathcal{P}(\bar{x})$ is equal to a label in Q or two leaf labels $\mathcal{P}(\bar{x})$ are equal and thus satisfy a join condition.

A valuation ν from Q to $\widehat{\mathcal{P}}$ is *relevant* to Q if the probability that Q is satisfied by ν is not zero, that is, $\Pr(\mathcal{P}, \nu \models Q) > 0$. We note that a relevant valuation satisfies Q with probability 1.

PROPOSITION 4.4. *Let $\widehat{\mathcal{P}}$ be a skeleton, Q a TP query, and ν a valuation from Q to $\widehat{\mathcal{P}}$. Then either $\Pr(\mathcal{P}, \nu \models Q) = 1$ or $\Pr(\mathcal{P}, \nu \models Q) = 0$.*

PROOF. We translate the problem into one of numbers. Let $S_\nu = \{\bar{x} \mid \mathcal{P}(\bar{x}), \nu \models Q\}$. It follows from the definition of satisfaction that S_ν is an affine subspace of $\mathbb{R}^k$. Then either $\dim S_\nu = k$ or $\dim S_\nu < k$. In the first case, $\widehat{\Pr}(S_\nu) = 1$, in the second, $\widehat{\Pr}(S_\nu) = 0$. $\square$

A leaf v of $\widehat{\mathcal{P}}$ is *relevant* to Q, if $\nu(n) = v$ for some valuation ν that is relevant to Q. Because of Proposition 4.4, with probability 1, Q returns exactly the leaves relevant to Q and no others.

Due to the proposition, one can easily find the relevant nodes by replacing all *cont*-labels of $\widehat{\mathcal{P}}$ with fresh distinct constants $\bar{x}$ from $\mathbb{R}$ that occur neither in $\widehat{\mathcal{P}}$ nor in Q. Then a node is relevant if it is retrieved by Q over $\mathcal{P}(\bar{x})$.

We define the *p-subskeleton of $\widehat{\mathcal{P}}$ relevant to Q*, denoted $\widehat{\mathcal{P}}_{|Q}$, as the smallest subtree of $\widehat{\mathcal{P}}$ that contains all leaves relevant to Q. Since only the relevant nodes contribute to the query result, and all of them do so with probability 1, it follows that the problem of determining the distribution of $Q^\alpha(\widehat{\mathcal{P}})$ is the same as the one of determining the distribution of α being evaluated over $\widehat{\mathcal{P}}_{|Q}$.

PROPOSITION 4.5. *For a skeleton $\widehat{\mathcal{P}}$ and aggregate query Q^α, we have $Q^\alpha(\widehat{\mathcal{P}}) = \alpha(\widehat{\mathcal{P}}_{|Q})$.*

Since we assume that all relevant nodes are labeled with continuous distributions, we can determine the distribution of query answers as in the first case.

Aggregate Queries over Arbitrary p-Documents. As before, we consider $\widehat{\mathcal{P}}$ and Q^α, but now assume that $\widehat{\mathcal{P}}$ is an arbitrary p-document $\widehat{\mathcal{P}} \in \mathsf{PrXML}_{cont}^{cie,mux,det}$. We also assume that over any document $d \in \mathcal{D}$, the query Q retrieves only nodes that stem from *cont*-nodes in $\widehat{\mathcal{P}}$. This seems natural, since in an application it is more likely that all the measurements one aggregates are imprecise, instead of some being fully precise and others not.

Under this assumption, as shown earlier, the distribution of Q^α over the space of a skeleton $\widehat{\mathcal{P}}_i$ can be obtained from the distributions on the leaves, essentially by convolution. The full space, moreover, is the convex sum of the skeleton spaces (see Eq. (1) in Section 3.2). It follows that the distribution of Q^α over $\widehat{\mathcal{P}}$ is the convex sum of the distributions over the skeletons $\widehat{\mathcal{P}}_i$.

PROPOSITION 4.6. *Let Q^α be an aggregate query, let $\widehat{\mathcal{P}} \in \mathsf{PrXML}_{cont}^{cie,mux,det}$ and $\widehat{\mathcal{P}}_1, \ldots, \widehat{\mathcal{P}}_n$ be the skeletons of $\widehat{\mathcal{P}}$, each with probability p_i. Then*

$$Q^\alpha(\widehat{\mathcal{P}}) = p_1 \cdot Q^\alpha(\widehat{\mathcal{P}}_i) + \cdots + p_n \cdot Q^\alpha(\widehat{\mathcal{P}}_n).$$

PROOF. Holds, since $[\![\widehat{\mathcal{P}}]\!] = p_1 \cdot [\![\widehat{\mathcal{P}}_1]\!] + \cdots + p_n \cdot [\![\widehat{\mathcal{P}}_n]\!]$. $\square$

If, for instance, $\alpha = \mathsf{max}$, then each $Q^\alpha(\widehat{\mathcal{P}}_i)$ is a max-convolution of some label distributions f_j. While each convolution can involve at most linearly many f_j, there may be exponentially many skeletons. This raises the question whether there are shorter representations of $Q^\alpha(\widehat{\mathcal{P}})$, which avoid the exponential blowup. We will see in Section 7 that this is indeed the case for p-documents in $\mathsf{PrXML}^{mux,det}$.

4.4. Computational Problems

In the discrete case, we are interested in the following three problems for an aggregate query Q^α where the input parameters are a discrete p-document $\widehat{\mathcal{P}}$ with corresponding random document $\mathcal{P}$ and possibly a number c.

Membership. Given $c \in \mathbb{R}$, is c in the carrier of $Q^\alpha(\mathcal{P})$, that is, is $\Pr(Q^\alpha(\mathcal{P}) = c) > 0$?
Probability. Given a number c, compute $\Pr(Q^\alpha(\mathcal{P}) = c)$.
Moments. Compute the moment $\mathbb{E}(Q^\alpha(\mathcal{P})^k)$, where $\mathbb{E}$ is the expected value.

Membership and probability computation can be used to return to a user the distribution $Q^\alpha(\mathcal{P})$ of an aggregate query. Computing the entire distribution may be too costly or the user may prefer a *summary* of the distributions. For example, a user may want to know its expected value $\mathbb{E}(Q^\alpha(\mathcal{P}))$ and the variance $\mathrm{Var}(Q^\alpha(\mathcal{P}))$. In general the summary can be an arbitrary k-th moment $\mathbb{E}(Q^\alpha(\mathcal{P})^k)$ and the moment computation problem addresses this issue.[6]

In the continuous case, we are interested in the following three problems for an aggregate query Q^α, where the input parameters are a p-document $\widehat{\mathcal{P}}$ with corresponding random document $\mathcal{P}$ and possibly two rational numbers c_1, c_2.

Membership. Given $c_1 < c_2$, is there a nonzero probability that the aggregate value $Q^\alpha(\mathcal{P})$ falls into the interval $]c_1, c_2[$? That is, is $\Pr(Q^\alpha(\mathcal{P}) \in \,]c_1, c_2[) > 0$?

Probability. Given $c_1 < c_2$, compute the probability that $Q^\alpha(\mathcal{P}) \in \,]c_1, c_2[$.

Moments. Compute the moment $\mathbb{E}(Q^\alpha(\mathcal{P})^k)$, where $\mathbb{E}$ is the expected value.

In the following, we investigate these problems for the classes of *cie* documents and *mux-det* documents. For each class, we further distinguish between aggregate queries of the types SP, TP, and TPJ (in the discrete case only) with the functions min, count, sum, countd, and avg. We do not discuss max and top$K$ since they behave similarly as min. In the article we mainly speak about *data complexity*, when the input is a p-document and the query is fixed. Occasionally we also consider *combined complexity*, when both the p-document and the query are inputs of the problem.

5. PRINCIPLES

This section presents a number of important principles and preliminaries that are used later on to support the complexity results.

5.1. Functions in #P and FP#P

We recall here the definitions of some classical complexity classes (see, e.g., Papadimitriou [1994]) that characterize the complexity of aggregate functions on PrXML$_{cont}^{cie,mux,det}$. An $\mathbb{N}$-valued function f is in #P if there is a nondeterministic polynomial-time Turing machine T such that for every input w, the number of accepting runs of T is the same as $f(w)$. A function is in FP#P if it is computable in polynomial time using an oracle for some function in #P. Following Cohen et al. [2009], we say that a function is FP#P-hard if there is a polynomial-time *Turing reduction* (that is, a reduction with access to an oracle to the problem reduced to) from every function in FP#P to it. Hardness for #P is defined in a standard way using Karp (many-one) reductions. For example, the function that counts for every propositional 2-DNF formula the number of satisfying assignments is in #P and #P-hard [Provan and Ball 1983], hence #P-complete. We notice that the usage of Turing reductions in the definition of FP#P-hardness implies that any #P-hard problem is also FP#P-hard. Therefore, to prove FP#P-completeness it is enough to show FP#P-membership and #P-hardness. Note also that #P-hardness clearly implies NP-hardness.

Membership in FP#P *for* PrXML^{cie,mux,det}. We show that the probability computation and moment computation problems for PrXML^{cie} are in FP#P by proof techniques adopted from Grädel et al. [1998], which will imply the same result for the less general PrXML^{mux,det} model. Since our problems are different from the ones considered in

---

[6]The variance is the *central* moment of order 2; it is known that the central moment of order k can be tractably computed from the regular moments of order $\leqslant k$.

Grädel et al. [1998] and the authors presented only a brief sketch of their techniques, we now give details on how we prove $\mathsf{FP}^{\#\mathsf{P}}$ membership.

We say that an aggregate function α is *scalable* if for every multiset of values B, one can compute in polynomial time a natural number M such that for every sub-multiset B' of B, the product $M \cdot \alpha(B')$ is a natural number. The aggregate functions count and countd are obviously scalable. On the other hand, min, sum, and avg are not scalable since they may take negative values. In fact this is not a real issue for the bags of values labeling leaves of p-documents, since one can always start by transforming the initial p-document to the one where all leaves are nonnegative, and, therefore, one ensures scalability of min, sum, and avg for the transformed p-document.

PROPOSITION 5.1. *Let α be an aggregate function that is computable in polynomial time and Q^α an aggregate* TPJ^α *query. The following functions mapping p-documents from* $\mathsf{PrXML}^{cie,mux,det}$ *to rational numbers are in* $\mathsf{FP}^{\#\mathsf{P}}$:

(1) *for every $c \in \mathbb{Q}$ the function $\widehat{\mathcal{P}} \mapsto \mathsf{Pr}(Q^\alpha(\mathcal{P}) = c)$;*
(2) *(provided α is scalable) for every $k \geqslant 1$, the function $\widehat{\mathcal{P}} \mapsto \mathbb{E}(Q^\alpha(\mathcal{P})^k)$.*

We use generating Turing machines to prove Proposition 5.1. We say that a nondeterministic Turing machine is a *generating* machine if: (1) all runs produce an output; (2) all runs terminate either in an accepting state or a nonaccepting state. Let T be a generating Turing machine with alphabet Σ and $u \in \Sigma^*$. Then we denote by $T(u)$ the multiset of outputs of T produced upon input u by an accepting run where the multiplicity of an output is equal to the number of accepting runs.

The proof of Proposition 5.1 is based on the following property of generating Turing machines.

LEMMA 5.2. *Let T be a generating polynomial-time Turing machine with alphabet Σ and let $g \colon \Sigma^* \to \mathbb{N}$ be a function computable in polynomial time. Then*

$$f(u) := \sum_{w \in T(u)} g(w), \tag{2}$$

where $g(w)$ is summed as often as w occurs in $T(u)$, defines a function $f \colon \Sigma^* \to \mathbb{N}$ such that $f \in \#\mathsf{P}$.

PROOF. We extend the machine T to a machine T' in such a way that the number of accepting runs of T' for input u is exactly $f(u)$ as follows. The machine T' first calls T on u. When T reaches an accepting state with output w, then T' computes $g(w)$ and creates $g(w)$ nondeterministic accepting branches, each of which corresponds to an accepting run. □

We are now ready to prove the proposition.

PROOF OF PROPOSITION 5.1. We show that both functions can be computed in polynomial time using a $\#\mathsf{P}$-oracle.

Since *mux* and *det* nodes can be transformed into *cie* nodes in polynomial time, we can restrict ourselves to p-documents of PrXML^{cie}. We assume that the p-document $\widehat{\mathcal{P}}$ has event variables $x_1, \ldots, x_n$ with probabilities $\Delta(x_i)$, which are rational numbers. Let K be the product of the denominators of the $\Delta(x_i)$'s. We note that K can be computed in polynomial time and that $K \cdot \Delta(x_i)$ is a natural number for all $1 \leqslant i \leqslant n$.

Let us start with probability computation. Let c be an arbitrary rational value. We show there is a $\#\mathsf{P}$-oracle that computes $f(\widehat{\mathcal{P}}) := K^n \cdot \mathsf{Pr}(Q^\alpha(\mathcal{P}) = c)$. Then the desired probability is obtained by dividing $f(\widehat{\mathcal{P}})$ by K^n, which can clearly be done in polynomial time. By Lemma 5.2 it is sufficient to exhibit a generating Turing machine

T and a polynomial time function g so that f can be represented as in (2). Now, T works as follows. For the input $\widehat{\mathcal{P}}$, it (1) computes K and writes it on the tape; (2) nondeterministically generates a truth assignment ν for the event variables of $\widehat{\mathcal{P}}$ and writes it on the tape; (3) computes $Q^\alpha(d)$ for the document d corresponding to the assignment ν (this can be done in polynomial time); and (4) if $Q^\alpha(d) = c$, then accepts, else does not accept. The function g computes the probability of $\Pr(\nu)$ and multiplies it by K^n, where n is the number of variables of the truth assignment ν. By definition of T and g, $K^{-n} \sum_{w \in T(\widehat{\mathcal{P}})} g(w) = \Pr(\alpha(\mathcal{P}) = c)$, which concludes the proof for the first claim of the proposition.

Assume now α scalable. We modify the previous construction as follows. In step (1), T also computes a proper M scalability factor for the multiset of values appearing in $\widehat{\mathcal{P}}$, step (3) writes $Q^\alpha(d)$ on the tape, and in (4) all documents d are accepted. The function g computes $K^n \cdot \Pr(\nu) \cdot M^k \cdot Q^\alpha(d)^k$, which is a natural number. Then clearly, $K^{-n} M^{-k} \sum_{w \in T(\widehat{\mathcal{P}})} g(w) = \mathbb{E}(\alpha(\mathcal{P})^k)$, which concludes the proof for the second claim of the proposition. □

We use Proposition 5.1 to show the following upper bound.

THEOREM 5.3. *The probability and moment computation problems for the class of aggregate* TPJ *queries over p-documents of* PrXML^{cie,mux,det} *are in* FP$^{\#P}$ *for aggregate functions* count, min, countd, sum, *and* avg.

PROOF. As already noted, count and countd are scalable so we can apply Proposition 5.1 as is. Furthermore, since min, sum, avg are polynomial time, the probability computation problem is in FP$^{\#P}$.

Let Q^α be an aggregate TPJ$^\alpha$ query. Suppose $\alpha = $ min or $\alpha = $ avg. Observe that the proof of Proposition 5.1 still works when the aggregate function varies from p-document to p-document, as long as it is polynomial time and scalable *for the values occurring* in the p-document that is being queried. For a p-document $\widehat{\mathcal{P}}$, let $C_\mathcal{P}$ be the lower negative value labeling $\widehat{\mathcal{P}}$ (or 0 if there is no such value). We define an aggregate function $\alpha_\mathcal{P}$ as $\alpha_\mathcal{P}(B) = \alpha(B) - C_\mathcal{P}$. Observe that $\alpha_\mathcal{P}$ is scalable for the values of $\widehat{\mathcal{P}}$. Therefore, one can compute $\mathbb{E}(Q^{\alpha_\mathcal{P}}(\mathcal{P})^k)$ in FP$^{\#P}$. Now,

$$\mathbb{E}(Q^\alpha(\mathcal{P})^k) = \mathbb{E}\left((Q^{\alpha_\mathcal{P}}(\mathcal{P}) + C_\mathcal{P})^k\right) = \sum_{j=0}^{k} \binom{k}{j} C_\mathcal{P}^{k-j} \mathbb{E}(Q_\mathcal{P}^\alpha(\mathcal{P})^j).$$

Since there are polynomially many terms that are each computable in polynomial time with a #P oracle, the whole expression is computable in polynomial time with a #P oracle.

Now consider the last case $\alpha = $ sum. For a document $\widehat{\mathcal{P}}$, we introduce the aggregate function avgcount$_\mathcal{P}$ defined by avgcount$_\mathcal{P}(B) = $ avg$_\mathcal{P}(B) \cdot$ count(B), which is scalable for the values of $\widehat{\mathcal{P}}$. We conclude by noting that

$$\mathbb{E}(Q^{\mathsf{sum}}(\mathcal{P})^k) = \mathbb{E}\left((Q^{\mathsf{avg}}(\mathcal{P}) \cdot Q^{\mathsf{count}}(\mathcal{P}))^k\right) = \mathbb{E}\left(\left((Q^{\mathsf{avg}_\mathcal{P}}(\mathcal{P}) + C_\mathcal{P}) \cdot Q^{\mathsf{count}}(\mathcal{P})\right)^k\right)$$

$$= \sum_{j=0}^{k} \binom{k}{j} C_\mathcal{P}^{k-j} \mathbb{E}(Q^{\mathsf{avgcount}_\mathcal{P}}(\mathcal{P})^j). \qquad \square$$

We now extend Proposition 5.1 from the discrete case to the continuous one.

Membership in FP$^{\#P}$ *for* PrXML$^{cie,mux,det}_{cont}$. In the discrete case scalability of aggregate functions guarantees the FP$^{\#P}$-membership, in the continuous case we need additional constraints on the scalability of p-documents themselves.

A p-document $\widehat{\mathcal{P}} \in \mathsf{PrXML}_{cont}^{cie,mux,det}$ is *p-scalable for* α (where *p* stands for *probability*) if for every two rational numbers $c_1 \leqslant c_2$ one can compute in polynomial time a natural number M, such that $M \cdot \mathsf{Pr}(\alpha(\mathcal{P}) \in [c_1, c_2])$ is a natural number. Moreover, $\widehat{\mathcal{P}}$ is *m-scalable for* α (where "m" stands for *moments*) if for every $k \in \mathbb{N}$ one can compute in polynomial time a natural number M, such that $M \cdot \mathbb{E}^k(\alpha(\mathcal{P}))$ is a natural number.

One can easily check that $\mathsf{PrXML}_{cont}^{cie,mux,det}$ p-documents labeled with distributions from $\mathsf{PP}(K)$ with *nonnegative* values are p-scalable and m-scalable for min, sum, and avg. We conclude with a lemma that gives $\mathsf{FP}^{\#\mathsf{P}}$-membership for probability and moment computation.

PROPOSITION 5.4. *Let α be one of* min, sum, avg *and* Q^α *an aggregate* TP^α *query. Let* $\widehat{\mathcal{P}} \in \mathsf{PrXML}_{cont}^{cie,mux,det}$ *be a p-document labeled with distributions from* $\mathsf{PP}(K)$ *for a fixed natural number K.*

(1) *If $\widehat{\mathcal{P}}$ is p-scalable for α, then the following function mapping p-documents to rational numbers is in* $\mathsf{FP}^{\#\mathsf{P}}$:

$$\text{for every rational } c_1 < c_2 \text{ the function } \widehat{\mathcal{P}} \mapsto \mathsf{Pr}(Q^\alpha(\mathcal{P}) \in [c_1, c_2]).$$

(2) *If $\widehat{\mathcal{P}}$ is m-scalable for α, then the following function mapping p-documents to rational numbers is also in* $\mathsf{FP}^{\#\mathsf{P}}$:

$$\text{for every } k \geqslant 1, \text{ the function } \widehat{\mathcal{P}} \mapsto \mathbb{E}(Q^\alpha(\mathcal{P})^k).$$

The proof is similar to the discrete case.

PROOF SKETCH. Again we assume $\widehat{\mathcal{P}} \in \mathsf{PrXML}_{cont}^{cie}$ without loss of generality. We first show the theorem for $\alpha \in \{\mathsf{min}, \mathsf{sum}\}$ and then discuss how the result can be extended to $\alpha = \mathsf{avg}$.

We start with two preliminary observations. We assume that the p-document $\widehat{\mathcal{P}}$ has skeletons $\widehat{\mathcal{P}}_i$ with probabilities p_i for $i = 1 \ldots t$, where every $\widehat{\mathcal{P}}_i$ can be generated by some truth assignment ν of the event variables occurring in $\widehat{\mathcal{P}}$ and p_i is the probability of this ν. Let N be the product of the denominators of the p_i. The number N can be computed in polynomial time from the probabilities of the variables that occur in $\widehat{\mathcal{P}}$ and $N \cdot p_i$ is a natural number for all i.

Let M_1 and M_2 be the constants for $\widehat{\mathcal{P}}$ that are used to define p-scalability and m-scalability of $\widehat{\mathcal{P}}$, respectively. That is, the products $M_1 \cdot \mathsf{Pr}(Q^\alpha(\mathcal{P}_i) \in [c_1, c_2])$ and $M_2 \cdot \mathbb{E}^k(Q^\alpha(\mathcal{P}_i))$ are natural numbers. Note that both products are computable in polynomial time. Indeed, by the definition of p-scalability, M_1 can be computed in polynomial time. Moreover, the probability $\mathsf{Pr}(Q^\alpha(\mathcal{P}_i) \in [c_1, c_2])$ can be computed in polynomial time for p-documents labeled with distributions in $\mathsf{PP}(K)$ as follows: (a) Replace the continuous distributions that label the leaves of $\widehat{\mathcal{P}}_i$ with fresh constants (that do not occur in $\widehat{\mathcal{P}}_i$), assume this yields a regular document d, and then query d with Q to retrieve *cont* nodes $l_1, \ldots, l_q$. (b) Compute $\int_{[c_1, c_2]}(f_{l_1} *_\alpha \cdots *_\alpha f_{l_q})$. One can see that $\mathsf{Pr}(Q^\alpha(\mathcal{P}_i) \in [c_1, c_2])$ is equal to the latter integral. The steps (a) and (b) are obviously polynomial, while polynomiality of the step (c) holds since the functions f_{l_i} are from $\mathsf{PP}(K)$. Analogously, one can show that the computation of $M_2 \cdot \mathbb{E}^k(Q(\mathcal{P}_i))$ is polynomial.

Now to prove the claim about probabilities, due to Lemma 5.2, it is sufficient to show that for any two $c_1 < c_2$, there is a $\#\mathsf{P}$ oracle that computes $f(\widehat{\mathcal{P}}) = \sum_{i=1}^{t}((N \cdot p_i) \cdot (M_1 \cdot \mathsf{Pr}(Q^\alpha(\mathcal{P}_i) \in [c_1, c_2])))$. For the claim about moments, the oracle should compute $f(\widehat{\mathcal{P}}) = \sum_{i=1}^{t}((N \cdot p_i) \cdot (M_2 \cdot \mathbb{E}^k(Q^\alpha(\mathcal{P}_i))))$. We proceed as in the proof of Proposition 5.1.

Since avg is not a monoid function, we cannot directly apply the preceding reasoning to the case when $\alpha = \mathsf{avg}$. At the same time, the proof uses computation of probabilities
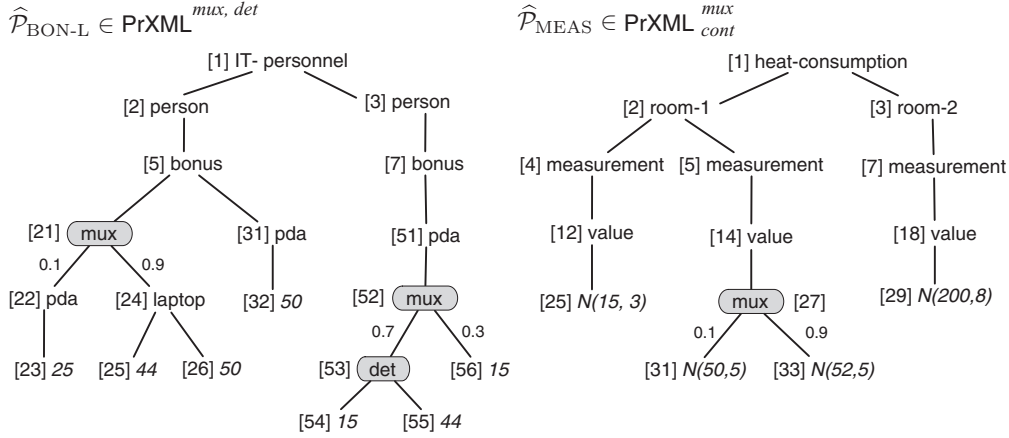
$\widehat{\mathcal{P}}_{\text{BON-L}} \in$ PrXML$^{mux,\,det}$ $\widehat{\mathcal{P}}_{\text{MEAS}} \in$ PrXML$^{mux}_{cont}$

Fig. 6.   p-Documents $\widehat{\mathcal{P}}_{\text{BON-L}}$ (left) and $\widehat{\mathcal{P}}_{\text{MEAS}}$ (right).

and moments only for skeletons, where the number of leaves in every document represented by a skeleton is the same as in the skeleton. This allows us to adapt the previous proof by using convolutions with respect to sum for determining the probability that the sum of aggregated nodes falls into $[nc_1, nc_2]$, where n is the fixed count of selected nodes in the skeleton. Similarly, to compute the moment of the avg query in a given skeleton, one computes the moment for the sum query and divides the result by the number of selected nodes.   □

5.2. Single-Pattern Query Evaluation and Document Aggregation

In this section, we show that computing the answer to an aggregate SP query and aggregating all leaves of a p-document are essentially the same operation.

We first show that for answering aggregate SP queries it is possible to isolate aggregation from query processing.

Let $\widehat{\mathcal{P}}$ be in PrXML^{cie,mux,det}. If Q is an SP query, we can apply it naïvely to $\widehat{\mathcal{P}}$, ignoring the distributional nodes. The result $\widehat{\mathcal{P}}_Q$ is the subtree of $\widehat{\mathcal{P}}$ containing the original root and as leaves the nodes satisfying Q (i.e., the nodes matched by the free variable of Q). It turns out that for all aggregate functions α, evaluating Q^α over $\widehat{\mathcal{P}}$ is the same as applying α to $\widehat{\mathcal{P}}_Q$. Therefore, answering an aggregate SP query Q^α over $\widehat{\mathcal{P}}$ in PrXML^{cie,mux,det} can be done in two steps: first one queries $\widehat{\mathcal{P}}$ with the nonaggregate part Q, which results in a p-document $\widehat{\mathcal{P}}_Q$, and then one aggregates *all* the leaves of $\widehat{\mathcal{P}}_Q$. As an illustration, the p-document $\widehat{\mathcal{P}}_{\text{BON-L}}$ from Figure 6 (left), where BON stands for *bonus*, is obtained by naïvely matching the query Q_{Bonus} from Figure 2 on the p-document $\widehat{\mathcal{P}}_{\text{PER-L}}$ from Figure 4.

For continuous p-documents $\widehat{\mathcal{P}} \in$ PrXML$^{cie,mux,det}_{cont}$ one first replaces each label that is a continuous distribution with a fresh constant (that does not appear anywhere else in the p-document); this yields a PrXML^{cie,mux,det} p-document $\widehat{\mathcal{P}}'$. Then one applies the same algorithm for computing $\widehat{\mathcal{P}}'_Q$ as described before and restores in $\widehat{\mathcal{P}}'_Q$ continuous labels on the leaves that remained from $\widehat{\mathcal{P}}'$, which results in a p-document $\widehat{\mathcal{P}}_Q$. Again, aggregation of all leaves of $\widehat{\mathcal{P}}_Q$ with α returns the same result as if the original p-document $\widehat{\mathcal{P}}$ was queried with Q^α. The document $\widehat{\mathcal{P}}_{\text{MEAS}}$ from Figure 6 (right), where MEAS stands for *measurements*, is obtained from $\widehat{\mathcal{P}}_{\text{CONS}}$ from Figure 5 and the single-path query that retrieves all children of a value node (i.e., in XPath notation, `//value/text()`).

The previous discussion leads to the following result, that can be formally proved by constructing the distributions $Q^\alpha(\widehat{\mathcal{P}})$ and $\alpha(\widehat{\mathcal{P}}_Q)$.

PROPOSITION 5.5. *Let $Q[n]$ be a nonaggregate* SP *query. Then for every p-document* $\widehat{\mathcal{P}} \in \mathsf{PrXML}_{cont}^{cie,mux,det}$ *we can compute in time polynomial in $|Q| + |\widehat{\mathcal{P}}|$ a p-subdocument* $\widehat{\mathcal{P}}_Q$ *of $\widehat{\mathcal{P}}$ such that, for every aggregate function α:*

$$Q^\alpha(\widehat{\mathcal{P}}) = \alpha(\widehat{\mathcal{P}}_Q).$$

In other words, evaluating an aggregate SP query is not harder than aggregating p-documents. We actually have some form of reciprocal to this result. A single-path query is said to be *trivial* if it is a root-only query. Evaluating any nontrivial aggregate query is as hard as aggregating p-documents.

PROPOSITION 5.6. *Let $\widehat{\mathcal{P}}$ be a p-document. Then for every nontrivial nonaggregate* SP *query $Q[n]$ we can compute in time polynomial in $|Q| + |\widehat{\mathcal{P}}|$ a p-document $\widehat{\mathcal{P}}^Q$ that uses the same kinds of distributional nodes as $\widehat{\mathcal{P}}$ and such that, for every aggregate function α:*

$$\alpha(\widehat{\mathcal{P}}) = Q^\alpha(\widehat{\mathcal{P}}^Q).$$

PROOF. We construct $\widehat{\mathcal{P}}^Q$ as follows. Assume the k nodes of Q are $n_1 \dots n_k$ in that order, with n_k the node to be aggregated. First, we construct a chain of nodes $u_1, u_2 \dots u_{k-1}$ such that the label of each u_i is compatible with that of n_i. The node u_1 is the root of $\widehat{\mathcal{P}}^Q$, and for $1 \leqslant i \leqslant k-2$, each u_i has a single child u_{i+1}. We add the whole $\widehat{\mathcal{P}}$ tree as child of u_{k-1}. Then, we remove all non-leaf ordinary nodes that were in $\widehat{\mathcal{P}}$, connecting leaves and distributional nodes that were children of an ordinary node to the closest remaining ancestor (possibly u_{k-1}). Then the nodes matched by $Q[n]$ in the resulting $\widehat{\mathcal{P}}^Q$ are exactly the ones that were leaves of $\widehat{\mathcal{P}}$, and $\alpha(\widehat{\mathcal{P}}) = Q^\alpha(\widehat{\mathcal{P}}^Q)$ for any aggregate function α. □

5.3. Hardness Results for Branching Queries

We now show a number of hardness results for queries in TP and TPJ that will provide general lower bounds.

With the next lemma, we can translate worst-case complexity results for nonaggregate queries to lower bounds of the complexity of computing probabilities of aggregate values and moments of distributions. An aggregate function α is *faithful* if there exists some value c such that $\alpha(\{\!| c |\!\}) \neq \alpha(\{\!| |\!\})$.

LEMMA 5.7. *Let Q be a* TPJ *query, $\widehat{\mathcal{P}}$ a p-document in* $\mathsf{PrXML}^{cie,mux,det}$, *and α a faithful aggregate function, with c such that $\alpha(\{\!| c |\!\}) \neq \alpha(\{\!| |\!\})$. Then one can construct in linear time an aggregate* TPJ *query Q'^α and a p-document $\widehat{\mathcal{P}}'$ such that for any $k \geqslant 1$,*

$$\mathsf{Pr}(\mathcal{P} \models Q) = \mathsf{Pr}\left(Q'^\alpha(\mathcal{P}') = \alpha(\{\!| c |\!\})\right) = \left(\mathbb{E}(Q'^\alpha(\mathcal{P}')^k) - \alpha(\{\!| |\!\})\right) / \left(\alpha(\{\!| c |\!\}) - \alpha(\{\!| |\!\})\right).$$

Moreover,

(1) *if $Q \in$ TP, then $Q'^\alpha \in$ TP$^\alpha$;*
(2) *if $\widehat{\mathcal{P}} \in \mathsf{PrXML}^{cie}$, then $\widehat{\mathcal{P}}' \in \mathsf{PrXML}^{cie}$;*
(3) *if $\widehat{\mathcal{P}} \in \mathsf{PrXML}^{mux,det}$, then $\widehat{\mathcal{P}}' \in \mathsf{PrXML}^{mux,det}$.*

PROOF. The construction of $\widehat{\mathcal{P}}'$ and Q'^α is illustrated on Figure 7 (left). If v is the root of $\widehat{\mathcal{P}}$, then one extends $\widehat{\mathcal{P}}$ with three extra nodes: a parent of v labeled root, a sibling labeled a, where a is a fresh constant that does not occur in neither $\widehat{\mathcal{P}}$ nor Q, and a child of the a node labeled c. The query Q is extended analogously, with the difference that the child of a is labeled with the aggregate variable x. The statement of the lemma
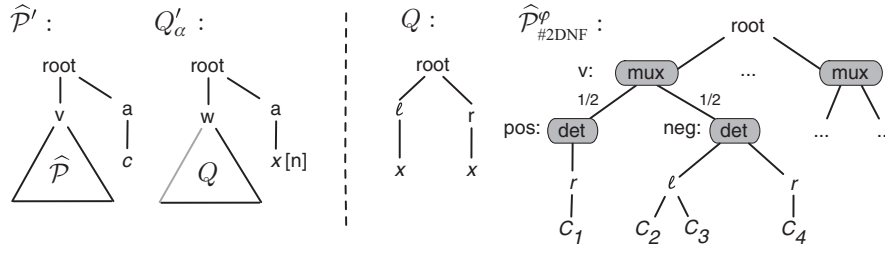
Fig. 7. Left: p-document $\widehat{\mathcal{P}}'$ and query Q'_α for reduction from query evaluation to probability and moments computation (Lemma 5.7). Right: Query $Q \in$ TPJ and p-document $\widehat{\mathcal{P}}_{\#2\text{DNF}}$ that show #P-hardness of TPJ over PrXML^{mux,det} (Lemma 5.8).

immediately follows from the observation that $Q'^\alpha(\mathcal{P}') = \alpha(\{\!\!\{c\}\!\!\})$ holds if and only if $\mathcal{P} \models Q$. □

Obviously, all considered aggregate functions are faithful. Note that the expression of the expected value makes this reduction useless if one allows ∞ as possible value for $\alpha(\{\!\!\{\}\!\!\})$, such as for min. However, it is easy to modify the construction of $\widehat{\mathcal{P}}'$ to obtain a similar reduction in the case of min also.

In Kimelfeld et al. [2008] it was shown that for every nontrivial Boolean tree-pattern query, computing the probability to match *cie* documents is #P-hard. By reducing #2-DNF, we can show that for the more restricted case of *mux-det* documents, evaluation of tree-pattern queries with joins can be #P-hard as well. We prove this here by giving an example of a hard TPJ query; note that we extend this result in Kharlamov et al. [2011] by showing that every query with a single join that is not equivalent to a join-free query is #P-hard.

LEMMA 5.8. *There is a Boolean* TPJ *query with* #P-*hard data complexity over* PrXML^{mux,det}.

PROOF. By reduction from propositional #2-DNF which is #P-hard. We illustrate how to construct the p-document $\widehat{\mathcal{P}}^\varphi_{\#2\text{DNF}}$ from a 2-DNF formula φ using the example: $\varphi = (w_1 \wedge v) \vee (\neg v \wedge w_2) \vee (\neg v \wedge w_3) \vee (w_4 \wedge \neg v)$, where v is a propositional variable and w_i for $i = 1 \ldots 4$ are some literals. For technical reasons we denote the clauses of φ as $C_1 = (w_1 \wedge v)$, $C_2 = (\neg v \wedge w_2)$, $C_3 = (\neg v \wedge w_3)$ and $C_4 = (w_4 \wedge \neg v)$. The construction can be easily extended to arbitrarily 2-DNF formulas.

Then $\widehat{\mathcal{P}}^\varphi_{\#2\text{DNF}}$ has the root with one *mux* child for every variable occurring in φ. In Figure 7 (right) we present a fragment of $\widehat{\mathcal{P}}^\varphi_{\#2\text{DNF}}$ corresponding to v. Under the *mux* node corresponding to v there are two uniformly distributed *det* children: one collects all positive occurrences of v in φ and the other one all the negative occurrences. In Figure 7 these nodes have pos and neg markers, respectively. The neg *det* child has two children l and r, under which we, respectively, list the clauses where $\neg v$ occurs on the left, that is, C_2 and C_3, and on the right, that is, C_4. Note that since every clause of φ is a conjunction of two literals, each variable always occurs either on the left or on the right position in a clause.

The query Q is the same for every φ in 2-DNF and presented in Figure 7 (right). Since every document of $[\![\widehat{\mathcal{P}}^\varphi_{\#2\text{DNF}}]\!]$ has the same probability, say p, one can see that the probability $\Pr(\widehat{\mathcal{P}}^\varphi_{\#2\text{DNF}} \models Q)$ is the number of satisfying assignment for φ times p.

Indeed, the query has the same root as $\widehat{\mathcal{P}}^\varphi_{\#2\text{DNF}}$, with two children l and r, that both have one child labeled with a join variable x. It is easy to see that every document $d \in [\![\widehat{\mathcal{P}}^\varphi_{\#2\text{DNF}}]\!]$ that satisfies Q has at least two leaves labeled with the same clause, say C, and the parents of the leaves are distinct labels from $\{l, r\}$. Hence, d corresponds to

Table II. Data Complexity of Query Evaluation over PrXML^{cie}

PrXML^{cie}	Aggregate query language			
	SP		TP	TPJ
Membership	NP-complete		NP-complete	NP-complete
Probability	FP$^{\#P}$-complete		FP$^{\#P}$-complete	FP$^{\#P}$-complete
Moments	count, sum	in P	FP$^{\#P}$-complete	FP$^{\#P}$-complete
	others	FP$^{\#P}$-complete		

a satisfying assignment μ of φ that makes C true. At the same time, for every such μ there is a document in $[\![\widehat{\mathcal{P}}^{\varphi}_{\#2\text{DNF}}]\!]$ satisfying Q. This yields a bijection χ between the set M of satisfying assignments for φ and the subset of $[\![\widehat{\mathcal{P}}^{\varphi}_{\#2\text{DNF}}]\!]$ whose documents satisfy Q. Since every document in $[\![\widehat{\mathcal{P}}^{\varphi}_{\#2\text{DNF}}]\!]$ has the same probability, say p, we conclude,

$$\text{Pr}(\widehat{\mathcal{P}}^{\varphi}_{\#2\text{DNF}} \models Q) = \sum_{\mu \in M} \text{Pr}(\chi(\mu)) = \sum_{\mu \in M} p = p|M|. \qquad \square$$

The result in Kimelfeld et al. [2008] and the previous lemma immediately yield the following complexity lower bounds for probability and moment computation for TP and TPJ.

COROLLARY 5.9. *For every aggregate function $\alpha \in \{\text{count}, \text{countd}, \text{min}, \text{sum}, \text{avg}\}$, there exist an aggregate* TP^{α} *query* Q_1^{α} *and an aggregate* TPJ^{α} *query* Q_2^{α}*, such that each of the following computation problems is* #P*-hard:*

(1) *probability computation for Q_1 over* PrXML^{cie}*;*
(2) *k-th moments of Q_1 over* PrXML^{cie}*, for any $k \geqslant 1$;*
(3) *probability computation for Q_2 over* PrXML^{mux,det}*;*
(4) *k-th moments of Q_2 over* PrXML^{mux,det}*, for any $k \geqslant 1$.*

This concludes the preliminaries. We are now ready to present our results on the complexity of aggregating PrXML^{cie} p-documents.

6. AGGREGATING P-DOCUMENTS WITH EVENT VARIABLES

We now study the problems introduced in Section 3 for the most general class of discrete p-documents, PrXML^{cie}, and then show how to extend these results to the continuous case of PrXML$^{cie}_{cont}$. By definition, one approach to deal with PrXML^{cie} is to first construct the entire px-space of a p-document $\widehat{\mathcal{P}}$, then to apply the aggregate query Q^{α} to each document in $[\![\widehat{\mathcal{P}}]\!]$ separately, and finally combine the results to obtain the distribution $Q^{\alpha}(\widehat{\mathcal{P}})$. This approach is expensive, since the number of possible documents is exponential in the number of variables occurring in $\widehat{\mathcal{P}}$.

Our complexity results show that for practically all functions and all problems nothing can be done that would be significantly more efficient. Decision problems are NP-complete while computational problems are FP$^{\#P}$-complete. The only exception is the computation of moments for aggregate single-path queries with sum and count. The intractability is due to dependencies between nodes of p-documents expressed using variables. We note that these intractability results hold for all nontrivial aggregate single-path queries.

Table II gives an overview of the data complexity results for the discrete PrXML^{cie} model, which we prove next.
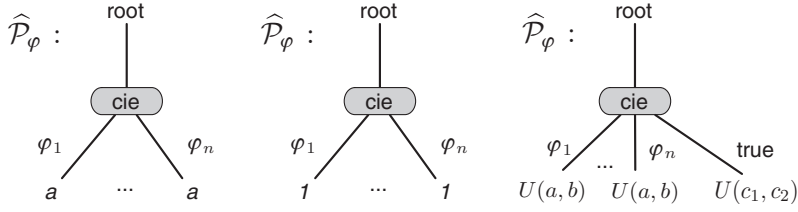
Fig. 8. Left: p-document $\widehat{\mathcal{P}}_\varphi$ for reduction from probability computation for CNF formulas to probability computation (Lemma 6.1). Center: p-document $\widehat{\mathcal{P}}_\varphi$ for reduction from probability computation for DNF formulas to moments computation (Lemma 6.4). Right: p-document $\widehat{\mathcal{P}}_\varphi$ for reduction from DNF falsifiability to membership for sum over continuous $\mathsf{PrXML}^{cie}_{cont}$ (Lemma 6.8).

6.1. Membership and Probability Computation for PrXML^{cie}

We first show that the membership and probability computation problems are hard over PrXML^{cie} for every nontrivial SP^α query. We show this result for a general class of aggregate functions that subsume all functions mentioned in this article.

Let α be an aggregate function over A. We say that α *distinguishes bags of $a \in A$ from the empty bag*, or simply *distinguishes bags of a*, if for every natural number n,

$$\alpha(\{\!\!\{\,\}\!\!\}) \neq \alpha(\{\!\!\{a^1, \ldots, a^n\}\!\!\}),$$

where the bag $\{\!\!\{a^1, \ldots a^n\}\!\!\}$ contains n occurrences of the single element a. All aggregate functions we considered in the article, namely, sum, count, min, avg, and countd distinguish bags of a, for any nonzero rational a.

LEMMA 6.1. *Let α be an aggregate function that distinguishes bags of some element a and Q^α a nontrivial SP^α query. Then for every propositional formula φ in DNF, one can compute in polynomial time a p-document $\widehat{\mathcal{P}}_{\varphi,\alpha} \in \mathsf{PrXML}^{cie}$ such that the following holds:*

$$\mathsf{Pr}\big(Q^\alpha(\mathcal{P}_{\varphi,\alpha}) = \alpha(\{\!\!\{\,\}\!\!\})\big) = 1 - \mathsf{Pr}(\varphi),$$

where $\mathsf{Pr}(\varphi)$ is the probability the formula φ holds given an independent probability distribution over variables of φ.

PROOF. Thanks to Proposition 5.6, we can restrict ourselves to the case of a query that aggregates all leaves of the document. Let $\varphi = \varphi_1 \vee \cdots \vee \varphi_n$, where each φ_i is a conjunction of literals. Then $\widehat{\mathcal{P}}_\varphi$ is constructed as follows: its root has a single *cie* child v, with in turn n children $v_1, \ldots, v_n$, where each v_i is labeled with a. The edge from v to v_i is labeled with φ_i. Thus, the event variables of $\widehat{\mathcal{P}}_\varphi$ are the variables of φ, with the same probability distribution. See $\widehat{\mathcal{P}}$ in Figure 8 (left).

The probability that Q^α returns $\alpha(\{\!\!\{\,\}\!\!\})$ (which is different from $\alpha(B)$ for any bag B of a's) is the probability that none of the φ_i's is true, that is, that φ is false. □

We obtain therefore the following result:

THEOREM 6.2. *Let α be an aggregate function that distinguishes bags of elements (in particular, α may be one of* count, countd, min, sum, avg*). Then, for every nontrivial query in SP^α:*

(1) *Membership over PrXML^{cie} is NP-hard.*
(2) *Probability computation over PrXML^{cie} is $\mathsf{FP}^{\#P}$-hard.*

PROOF. This is a direct consequence of the reduction of Lemma 6.1 and of the following two facts: (1) A formula has probability < 1 if and only if it is falsifiable, which is

NP-hard to check. (2) Computing the probability of a formula in CNF ($\neg\varphi$) is $\mathsf{FP}^{\#\mathsf{P}}$-hard [Papadimitriou 1994; Grädel et al. 1998].   □

Furthermore, we have already shown in Theorem 5.3 that probability computation over TPJ for all considered aggregate functions is in $\mathsf{FP}^{\#\mathsf{P}}$. Similarly, membership over TPJ is in NP for all considered aggregate functions since, given a query, guessing a world and evaluating the query takes no more than polynomial time.

6.2. Moments Computation for $\mathsf{PrXML}^{\textit{cie}}$

We show how to compute moments over PrXML^{cie}.

THEOREM 6.3. *Let $\alpha \in \{\mathsf{sum}, \mathsf{count}, \mathsf{min}, \mathsf{avg}, \mathsf{countd}\}$. Then computation of moments of any degree over PrXML^{cie} is in $\mathsf{FP}^{\#\mathsf{P}}$ for the class TPJ^{α}. Moreover, the problem is*

(1) *of polynomial combined complexity for the classes $\mathsf{SP}^{\mathsf{sum}}$ and $\mathsf{SP}^{\mathsf{count}}$;*
(2) *#P-hard for the classes $\mathsf{TP}^{\mathsf{sum}}$ and $\mathsf{TP}^{\mathsf{count}}$;*
(3) *#P-hard for any nontrivial query in the classes $\mathsf{SP}^{\mathsf{min}}$, $\mathsf{SP}^{\mathsf{avg}}$ and $\mathsf{SP}^{\mathsf{countd}}$.*

PROOF. Again, the $\mathsf{FP}^{\#\mathsf{P}}$ upper bound follows from Theorem 5.3. Claim 2 follows from Corollary 5.9. Claim 3 follows from Lemma 6.4, presented next, and from the fact that probability computation for DNF formulas is $\mathsf{FP}^{\sharp}$-hard [Grädel et al. 1998]. To prove Claim 1, we rely on Proposition 5.5, which reduces answering aggregate SP queries to evaluating aggregate functions, and the following two Lemmas 6.5 and 6.6. After we present the lemmas we finish the proof of the theorem.   □

We start with a supporting lemma for Claim 3 of the theorem.

LEMMA 6.4. *Let $\alpha \in \{\mathsf{min}, \mathsf{avg}, \mathsf{countd}\}$ and Q^{α} a nontrivial SP query. Then for every propositional DNF formula φ, one can compute in polynomial time a p-document $\widehat{\mathcal{P}}_{\varphi} \in \mathsf{PrXML}^{cie}$ such that $\mathbb{E}(Q^{\alpha}(\mathcal{P}_{\varphi})^{k}) = \mathsf{Pr}(\varphi)$ for any $k \geqslant 1$.*

PROOF. Again, using Proposition 5.6, it it suffices to show that probability computation of a DNF formula boils down to aggregating all leaves of a p-document.

Let $\varphi = \varphi_1 \vee \cdots \vee \varphi_n$ be a formula in DNF with n variables and m satisfying assignments. Consider $\widehat{\mathcal{P}}_{\varphi}$ (Figure 8, center) that has below its root a *cie* node v with children $v_1, \ldots, v_n$ that are leaves of $\widehat{\mathcal{P}}_{\varphi}$ and each v_i is labeled with 1. If α is min, we add an extra child v' to v, labeled with 0. The edges from v to each v_i are labeled with φ_i (the edge to v' for min is labeled with true).

The proof immediately follows from the observation that for every natural number m and bag $B = \{\!\!\{1^1, \ldots, 1^m\}\!\!\}$, that contains the single element 1 occurring m times, $\alpha(B) = 1$, and for $B = \{\!\!\{\}\!\!\}$ it holds $\alpha(B) = 0$.   □

The next lemma shows that the computation of the expected value for sum over a px-space, regardless whether it can be represented by a p-document, can be polynomially reduced to the computation of an auxiliary probability.

LEMMA 6.5 (REGROUPING SUMS). *Let $\mathcal{S}$ be a px-space and V the set of all leaves occurring in the documents of $\mathcal{S}$. Suppose that the function θ labels all leaves in V with rational numbers and let $\mathsf{sum}(\mathcal{S})$ be the random variable defined by sum on $\mathcal{S}$. Then*

$$\mathbb{E}(\mathsf{sum}(\mathcal{S})^k) = \sum_{(v_1,\ldots,v_k)\in V^k} \left(\prod_{i=1}^{k} \theta(v_i)\right) \times \mathsf{Pr}(\{d \in \mathcal{S} \mid v_1,\ldots,v_k \text{ occur in } d\}),$$

where the last term denotes the probability that a random document $d \in \mathcal{S}$ contains all the nodes $v_1, \ldots, v_k$.

PROOF. Intuitively, the proof exploits the fact that $\mathbb{E}(\mathsf{sum}(\mathcal{S}))$ is a sum over documents of sums over nodes, which can be rearranged as a sum over nodes of sums over documents. Formally:

$$\mathbb{E}(\mathsf{sum}(\mathcal{S})^k) = \sum_{d \in \mathcal{S}} \left(\sum_{\substack{v \in V \\ v \in d}} \theta(v) \right)^k \mathsf{Pr}(d) = \sum_{d \in \mathcal{S}} \left(\sum_{\substack{v_1 \in V \\ v_1 \in d}} \theta(v_1) \right) \left(\sum_{\substack{v' \in V \\ v' \in d}} \theta(v') \right)^{k-1} \mathsf{Pr}(d)$$

$$= \sum_{v_1 \in V} \theta(v_1) \sum_{\substack{d \in \mathcal{S} \\ v_1 \in d}} \left(\sum_{\substack{v' \in V \\ v' \in d}} \theta(v') \right)^{k-1} \mathsf{Pr}(d) = \dots$$

$$= \sum_{v_1 \in V} \theta(v_1) \cdots \sum_{v_k \in V} \theta(v_k) \sum_{\substack{d \in \mathcal{S} \\ v_1, \dots, v_k \in d}} \mathsf{Pr}(d)$$

$$= \sum_{(v_1, \dots, v_k) \in V^k} \left(\prod_{i=1}^{k} \theta(v_i) \right) \mathsf{Pr}(\{d \in \mathcal{S} \mid v_1, \dots, v_k \text{ occur in } d\}). \qquad \square$$

The auxiliary probability introduced in the previous lemma can be in fact computed in polynomial time for px-spaces represented by $\widehat{\mathcal{P}} \in \mathsf{PrXML}^{cie}$.

LEMMA 6.6. *There is a linear-time algorithm that computes, given a p-document $\widehat{\mathcal{P}} \in \mathsf{PrXML}^{cie}$ and leaves $v_1, \dots, v_k$ occurring in $\widehat{\mathcal{P}}$, the probability*

$$\mathsf{Pr}(\{d \in [\![\widehat{\mathcal{P}}]\!] \mid v_1, \dots, v_k \text{ occur in } d\}).$$

Let φ_i be the conjunction of all formulas that label the path from the root of $\widehat{\mathcal{P}}$ to v_i for $1 \leqslant i \leqslant k$. Then the probability considered is equal to $\mathsf{Pr}(\varphi_1 \wedge \cdots \wedge \varphi_k)$ and computable in linear time.

Now we are ready to conclude the proof of the theorem.

PROOF OF THEOREM 6.3, CLAIM 1. By Lemma 6.5, the k-th moment of sum over $\widehat{\mathcal{P}}$ is the sum of $|V|^k$ products, where V is the set of leaves of $\widehat{\mathcal{P}}$. The first term of each product, $\prod_{i=1}^{k} \theta(v_i)$, can be computed in time at most $|\widehat{\mathcal{P}}|^k$. By Lemma 6.6, the second term $\mathsf{Pr}(\{d \in [\![\widehat{\mathcal{P}}]\!] \mid v_1, \dots, v_k \text{ occur in } d\})$ can be computed in time linear in $\widehat{\mathcal{P}}$. This shows that for every $k \geqslant 1$, the k-th moment of sum can be computed in polynomial time. The claim for count follows as a special case, where all leaves carry the label 1. $\square$

6.3. Aggregating Continuous $\mathsf{PrXML}^{cie}_{cont}$

We now investigate under which conditions the results for discrete p-documents PrXML^{cie} extend to the continuous case. Table III gives an overview of the results that are proved next.

A function is *piecewise-continuous* if it is a finite union of continuous functions f with disjoint domains, where each f is defined over an interval over the reals, possibly right or left-closed. We denote the set of all piecewise-continuous functions as PC.

Membership. We now define classes of p-documents for which we can guarantee that membership can be checked in nondeterministic polynomial time.

We start with p-documents for which membership of min is in NP. Given a p-document $\widehat{\mathcal{P}} \in \mathsf{PrXML}_{cont}$ that is labeled with distributions in PC and two rational numbers $c_1 < c_2$, one can test $\mathsf{Pr}(\mathsf{min}(\widehat{\mathcal{P}}) \in [c_1, c_2]) > 0$ by examining the possible values of each leaf of $\widehat{\mathcal{P}}$.

Table III. Data Complexity of Query Evaluation over PrXML$_{cont}^{cie}$

PrXML$_{cont}^{cie}$	Aggregate query language		
		SP	TP
Membership		NP-complete*	NP-complete*
Probability		FP$^{\#P}$-complete†	FP$^{\#P}$-complete†
Moments	sum	in P†	FP$^{\#P}$-complete†
	min, avg	FP$^{\#P}$-complete†	

*NP-membership holds for min-reasonable (for TP^{min}) and sum-reasonable (for TP^{sum} and TP^{avg}) p-documents labeled with PC distributions.
†The upper bound holds for p-documents labeled with PP(K) distributions, for a fixed K.
In all cases lower bounds already hold for p-documents labeled with PP(0) distributions.

Intuitively, if for every leaf l of $\widehat{\mathcal{P}}$ the probability that the value of l is greater than c_1 is nonzero and for at least one leaf l the probability that the value of l falls in the interval $[c_1, c_2]$ is nonzero, then the minimum of values across all the leaves of $\widehat{\mathcal{P}}$ also falls in the interval $[c_1, c_2]$ with nonzero probability. A p-document $\widehat{\mathcal{P}} \in$ PrXML$_{cont}$ labeled with distributions in PC is min-*reasonable* if this test can be performed in polynomial time. Formally, let $l_1, \ldots, l_n$ be the leaves of $\widehat{\mathcal{P}}$, then $\widehat{\mathcal{P}}$ is min-reasonable if for all rational numbers $c_1 < c_2$ and integer $1 \leqslant i \leqslant n$, it can be decided in polynomial time whether

$$\int_{[c_1,\infty[} f_{l_i}(x)\,dx > 0; \qquad \text{and} \qquad \int_{[c_1,c_2]} f_{l_i}(x)\,dx > 0.$$

A p-document $\widehat{\mathcal{P}} \in$ PrXML$_{cont}^{cie,mux,det}$ is min-*reasonable* if *all* its skeletons are min-reasonable. For example, p-documents that are labeled with distributions in PP are min-reasonable.

We now discuss p-documents for which membership of sum is in NP. Consider again a p-document $\widehat{\mathcal{P}} \in$ PrXML$_{cont}$ labeled with distributions in PC. Since sum is a continuous function one can again test $\Pr(\text{sum}(\widehat{\mathcal{P}}) \in [c_1, c_2]) > 0$ by examining the possible values of each leaf of $\widehat{\mathcal{P}}$. Intuitively, if for every leaf l of $\widehat{\mathcal{P}}$ there is a value v_l such that the probability of every small interval around v_l is nonzero and the sum $s = \sum_l v_l$ across all these values falls in $[c_1, c_2]$, then every small enough interval around s falls into $[c_1, c_2]$, which guarantees that $\Pr(\text{sum}(\widehat{\mathcal{P}}) \in [c_1, c_2]) > 0$. A p-document $\widehat{\mathcal{P}} \in$ PrXML$_{cont}$ with leaves $l_1, \ldots, l_n$ labeled with distributions in PC is sum-*reasonable* if this test can be performed in polynomial time, formally, if for all rational numbers $c_1 < c_2$ it can be decided in polynomial time whether there is a vector of rationals $v_1, \ldots, v_n$ such that for all $\varepsilon > 0$, $\int_{(v_i-\varepsilon, v_i+\varepsilon)} f_{l_i}(x)\,dx > 0$ for $1 \leqslant i \leqslant n$ and $\sum_{i=1}^n v_i \in [c_1, c_2]$. Observe that p-documents labeled with distributions in PP are sum-reasonable.

THEOREM 6.7. *Let* $\alpha \in \{\text{min}, \text{sum}, \text{avg}\}$. *Then membership over* PrXML$_{cont}^{cie}$ *for p-documents that are labeled with distributions in* PC *is in* NP *for the class*

(1) TP^{min} *over* min-*reasonable* PrXML$_{cont}^{cie}$ *p-documents;*
(2) TP^{sum} *and* TP^{avg} *over* sum-*reasonable* PrXML$_{cont}^{cie}$ *p-documents.*

Moreover, membership is NP-*hard for every nontrivial aggregate query in* SP$^\alpha$, *for p-documents labeled with distributions in* PP(0).

In order to prove the NP-hardness result, we present now a lemma whose proof will be later used as a general principle for extending hardness results from discrete models to continuous ones.

LEMMA 6.8. *For every propositional DNF formula φ, and any two rational numbers $c_1 < c_2$, one can compute in polynomial time a p-document $\widehat{\mathcal{P}}_\varphi \in$ PrXML^{cie} labeled with distributions in* PP(0) *such that the following are equivalent:*

(1) φ *is falsifiable,*
(2) $\mathsf{Pr}(\mathsf{sum}(\mathcal{P}_\varphi) \in [c_1, c_2]) > 0$.

PROOF. Let $\varphi = \varphi_1 \vee \cdots \vee \varphi_n$, where each φ_i is a conjunction of literals. If $c_1 < 0$, then we pose $[a, b] = [2c_1 - c_2, c_1 - c_2]$. Otherwise, we pose $[a, b] = [c_2 - c_1, 2c_2 - c_1]$. Then $\widehat{\mathcal{P}}_\varphi$ is as in Figure 8, right. Each leaf that has an incident edge labeled with φ_i has for label the uniform distribution $U(a, b)$ over $[a, b]$. The label of the last child, v, with edge annotated with true is $U(c_1, c_2)$.

If in some world, v is the only leaf (i.e., if φ is falsifiable), then $\mathsf{Pr}(\mathsf{sum}(\mathcal{P}_\varphi) \in [c_1, c_2]) = 1$. Conversely, if φ is a tautology, either $\mathsf{Pr}(\mathsf{sum}(\mathcal{P}_\varphi) < c_1) = 1$ or $\mathsf{Pr}(\mathsf{sum}(\mathcal{P}_\varphi) > c_2) = 1$. □

PROOF SKETCH OF THEOREM 6.7. An NP decision procedure for TP$^{\mathsf{min}}$ and TP$^{\mathsf{sum}}$ starts by guessing one skeleton $\widehat{\mathcal{P}}_i$ of $\widehat{\mathcal{P}}$, then evaluates the query regarding *cont* nodes as constants, then substitutes the constant with the original *cont* nodes, and finally uses the min-reasonable or sum-reasonable character of $\widehat{\mathcal{P}}$ (and therefore of $\widehat{\mathcal{P}}_i$) to decide whether the aggregate query result falls into $[c_1, c_2]$ with positive probability. For TP$^{\mathsf{avg}}$, NP-membership can be proved as for TP$^{\mathsf{sum}}$, since in every document represented by a skeleton of a p-document the number of leaves is the same as in the skeleton.

NP-hardness can be shown for any nontrivial SP query over PP(0) documents, by first applying Lemma 6.8 and then Proposition 5.6 to reduce aggregation to the evaluation of an arbitrary nontrivial SP query. □

Probability Computation. We now discuss probability computation over PrXML$^{cie}_{cont}$.

THEOREM 6.9. *Let $\alpha \in \{$min, sum, avg$\}$ and K be a natural number. Then probability computation over* PrXML$^{cie}_{cont}$ *for p-documents that are labeled with distributions in* PP(0) *is* #P-*hard for every aggregate query in* SP$^\alpha$. *Moreover, probability computation is in* FP$^{\#P}$ *for* TP$^\alpha$ *and the class of p-documents that are labeled with distributions in* PP(K).

PROOF SKETCH. #P-hardness of probability computation can be shown by combining Proposition 5.6 with the same reduction as in the discrete case (see Lemma 6.1) using uniform probability distributions on leaves. For every DNF φ we construct a p-document $\widehat{\mathcal{P}}$ and an SP$^\alpha$ query Q^α such that $\mathsf{Pr}(Q^\alpha(\widehat{\mathcal{P}}) \in (-\varepsilon, +\varepsilon)) = 1 - \mathsf{Pr}(\varphi)$ for some $\varepsilon > 0$. FP$^{\#P}$-membership follows from Proposition 5.4. □

Moment Computation. We finally discuss moments over PrXML$^{cie}_{cont}$.

THEOREM 6.10. *Let $\alpha \in \{$min, sum, avg$\}$ and K be a natural number. Then computation of moments of any degree over* PrXML$^{cie}_{cont}$ *is in* FP$^{\#P}$ *for the class* TP$^\alpha$ *and the class of p-documents that are labeled with distributions in* PP(K). *Moreover the problem is*

(1) *of polynomial combined complexity for the class* SP$^{\mathsf{sum}}$ *and p-documents labeled with distributions in* PP(K)*;*
(2) #P-*hard for the class* TP$^{\mathsf{sum}}$*;*
(3) #P-*hard for any query in the classes* SP$^{\mathsf{min}}$ *and* SP$^{\mathsf{avg}}$.

PROOF SKETCH. FP$^{\#P}$-membership and Claims 2 and 3 are shown as in the discrete case, similarly to Theorem 6.9. Claim 1 follows from the following Lemma 6.11 and the fact that both the integral and the probability that occur in the formula of the lemma can be computed in polynomial time for p-documents labeled with PP(K) distributions. □

We now present a continuous version of the lemma on *regrouping sums*.

LEMMA 6.11. *Let $\mathcal{S}' = \sum_{i=1}^{n} \text{Pr}(\mathcal{S}_i) \cdot \mathcal{S}_i$ be a continuous px-space, where each $\mathcal{S}_i$ is a continuous px-space in which all documents have one skeleton with set of leaves $L(\mathcal{S}_i)$. Let $L(\mathcal{S})$ be the set of all leaf nodes occurring in $\mathcal{S}$. Suppose that the function v labels all leaves in L with real numbers according to the distribution attached to the leaf and let $\text{sum}_{\mathcal{S}'}$ be the random variable defined by* sum *on $\mathcal{S}'$. Then*

$$\mathbb{E}(\text{sum}_{\mathcal{S}'}^{k}) = \sum_{(l_{t_1},\ldots,l_{t_k}) \in L^k} \left(\int_{\substack{v(l_1),\ldots,v(l_n) \\ \{l_1,\ldots,l_n\}=L(\mathcal{S})}} \prod_{i=1}^{k} v(l_{t_i}) \prod_{j=1}^{n} f_{l_j}(v(l_j)) \, dv(l_1)\cdots dv(l_n) \times P \right),$$

where $P = \text{Pr}(\{\mathcal{S} \in \{\mathcal{S}_1, \ldots, \mathcal{S}_n\} \mid l_{t_1}, \ldots, l_{t_k} \text{ occur in } \mathcal{S}\})$ and denotes the probability that one of the structural worlds $\mathcal{S}$ of $\mathcal{S}'$ contains all nodes $l_1, \ldots, l_k$.

The proof is an extension of the one for Lemma 6.5.

7. AGGREGATION WITH MONOID FUNCTIONS

The previous section highlighted the inherent difficulty of computing aggregate queries over *cie* documents. The intuitive reason for this difficulty is that the event variables used in a p-document can impose constraints between the structure of subdocuments in very different locations. In contrast, *mux-det* documents only express "local" dependencies. As a consequence, for the special case of single-path queries and monoid aggregate functions, *mux-det* documents allow for a conceptually simpler computation of distributions, which in a number of cases is also computationally efficient.

7.1. Monoid Aggregates over Discrete Probabilistic Data

The key to developing methods in this setting is Proposition 5.5, which reduces the evaluation of a single-path aggregate query Q^α over $\widehat{\mathcal{P}}$ to the evaluation of the function α over the document $\widehat{\mathcal{P}}_Q$. Note that $\widehat{\mathcal{P}}_Q$ is again a *mux-det* document if $\widehat{\mathcal{P}}$ is one. Therefore, we can concentrate on the question of evaluating α over *mux-det* p-documents.

We are going to show how a *mux-det* p-document $\widehat{\mathcal{P}}$ can be seen as a recipe for constructing the px-space $[\![\widehat{\mathcal{P}}]\!]$ in a bottom-up fashion, starting from elementary spaces represented by the leaves and using essentially two kinds of operations, convex union and product. Convex union corresponds to *mux* nodes and product corresponds to *det* nodes and regular nodes. (To be formally correct, we would need to distinguish between two slightly different versions of product for *det* and regular nodes. However, to simplify our exposition, we only discuss the case of regular nodes and briefly indicate below the changes necessary to deal with *det* nodes.)

For any α, the distribution over the space described by a leaf of $\widehat{\mathcal{P}}$ is a Dirac distribution, that is, a distribution of the form δ_a, where $\delta_a(b) = 1$ if and only if $a = b$. For monoid functions α, the two operations on spaces, convex union and product, have as counterparts two operations on distributions, convex sum and convolution, by which one can construct the distribution $\alpha(\widehat{\mathcal{P}})$ from the Dirac distributions of the leaves of $\widehat{\mathcal{P}}$. We sketch in the following both the operations on spaces and on distributions, and the way in which they are related.

As the base case, consider a leaf node v with label l. This is the simplest p-document possible, which constitutes an elementary px-space that contains one document, namely node v with label l, and assigns the probability 1 to that document. Over this space, α evaluates with probability 1 to $\alpha(\{\!|l|\!\})$, hence, the probability distribution is $\delta_{\alpha(\{\!|l|\!\})}$. As a special case, if α is a monoid aggregation function over M, the distribution of α over the space containing only the empty document is $\delta_\perp$, where $\perp$ is the identity of M.

$$\widehat{\mathcal{P}}_v: \qquad \alpha(\widehat{\mathcal{P}}_v) = p_1\alpha(\widehat{\mathcal{P}}_1) + \cdots + p_n\alpha(\widehat{\mathcal{P}}_n) \qquad \alpha(\widehat{\mathcal{P}}_v) = \alpha(\widehat{\mathcal{P}}_1) *_M \cdots *_M \alpha(\widehat{\mathcal{P}}_n)$$
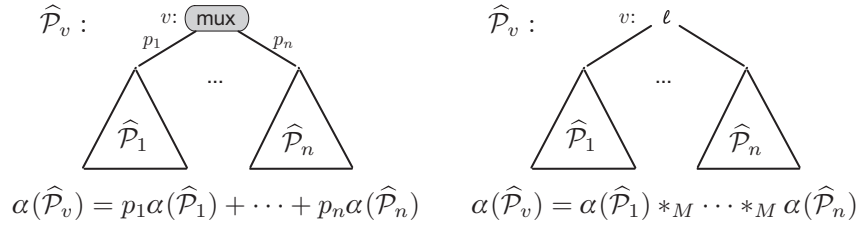
Fig. 9.   Distribution of monoid functions over composed PrXML^{mux,det} documents.

Inductively, suppose that v is a *mux*-node in $\widehat{\mathcal{P}}$, the subtrees below v are $\widehat{\mathcal{P}}_1, \ldots, \widehat{\mathcal{P}}_n$, and the probability of the i-th subtree $\widehat{\mathcal{P}}_i$ is p_i (see Figure 9, left). Without loss of generality we can assume that the p_i are convex coefficients, that is, $p_1 + \cdots + p_n = 1$, since we admit the empty tree as a special p-document.

Let $\widehat{\mathcal{P}}_v$ denote the subtree rooted at v. Then the semantics of *mux*-nodes implies that the px-space $[\![\widehat{\mathcal{P}}_v]\!] = (\mathcal{D}_v, \mathsf{Pr}_v)$ is the convex union of the spaces $[\![\widehat{\mathcal{P}}_i]\!] = (\mathcal{D}_i, \mathsf{Pr}_i)$. Indeed, (1) $\mathcal{D}_v$ is the disjoint union of the $\mathcal{D}_i$ (in other words, for any $d \in \mathcal{D}_v$, there is exactly one $\mathcal{D}_i$ such that $d \in \mathcal{D}_i$); (2) for any document $d \in \mathcal{D}_v$, we have that $\mathsf{Pr}_v(d) = p_i \cdot \mathsf{Pr}_i(d)$, where $d \in \mathcal{D}_i$.

As a consequence, $\alpha(\widehat{\mathcal{P}}_v)(c)$, the probability that α has the value c over $\widehat{\mathcal{P}}_v$, equals the weighted sum $p_1 \cdot \alpha(\widehat{\mathcal{P}}_1)(c) + \cdots + p_n \cdot \alpha(\widehat{\mathcal{P}}_n)(c)$ of the probabilities $\alpha(\widehat{\mathcal{P}}_i)(c)$ that α has the value c over $\widehat{\mathcal{P}}_1, \ldots, \widehat{\mathcal{P}}_n$. In a more compact notation we can write this as $\alpha(\widehat{\mathcal{P}}_v) = p_1 \cdot \alpha(\widehat{\mathcal{P}}_1) + \cdots + p_n \cdot \alpha(\widehat{\mathcal{P}}_n)$, which means that the distribution $\alpha(\widehat{\mathcal{P}}_v)$ is a convex sum of the $\alpha(\widehat{\mathcal{P}}_i)$.

For the second induction step, suppose that v is a regular non-leaf node in $\widehat{\mathcal{P}}$, with the label l (see Figure 9, right). Similar to the previous case, suppose that the subtrees below v are $\widehat{\mathcal{P}}_1, \ldots, \widehat{\mathcal{P}}_n$, that $[\![\widehat{\mathcal{P}}_v]\!] = (\mathcal{D}_v, \mathsf{Pr}_v)$ and that $[\![\widehat{\mathcal{P}}_i]\!] = (\mathcal{D}_i, \mathsf{Pr}_i)$ for $1 \leqslant i \leqslant n$. Moreover, the $\mathcal{D}_i$ are mutually disjoint.

Every document $d \in \mathcal{D}_v$ has as root the node v, which carries the label l, and subtrees $d_1, \ldots, d_n$, where $d_i \in \mathcal{D}_i$. We denote such a document as $d = v^l(d_1, \ldots, d_n)$. Conversely, according to the semantics of regular nodes in *mux-det* documents, every combination $\{d_1, \ldots, d_n\}$ of documents $d_i \in \mathcal{D}_i$ gives rise to an element $v^l(d_1, \ldots, d_n) \in \mathcal{D}_v$. (Note that, due to the mutual disjointness of the $\mathcal{D}_i$, the elements of $\mathcal{D}_v$ are in bijection with the tuples in the Cartesian product $\mathcal{D}_1 \times \cdots \times \mathcal{D}_n$.)

Consider a collection of documents $d_i \in \mathcal{D}_i$, $1 \leqslant i \leqslant n$, with probabilities $q_i := \mathsf{Pr}_i(d_i)$. Each d_i is the result of dropping some children of *mux* nodes in $\widehat{\mathcal{P}}_i$ and q_i is the product of the probabilities of the surviving children. Then $d := v^l(d_1, \ldots, d_n)$ is the result of dropping simultaneously the same children of those *mux* nodes, this time within $\widehat{\mathcal{P}}_v$. The set of surviving children in $\widehat{\mathcal{P}}_v$ is exactly the union of the sets of children having survived in each $\widehat{\mathcal{P}}_i$ and, consequently, for the probability $q := \mathsf{Pr}_v(d)$ we have that $q = q_1 \cdots q_n$. In summary, this shows that the probability space $(\mathcal{D}_v, \mathsf{Pr}_v)$ is structurally the same as the product of the spaces $(\mathcal{D}_i, \mathsf{Pr}_i)$.

Suppose now that, in addition, α is a monoid aggregate function taking values in $(M, \oplus, \perp)$. Then for any document $d = v^l(d_1, \ldots, d_n) \in \widehat{\mathcal{P}}_v$ we have that $\alpha(d) = \alpha(d_1) \oplus \cdots \oplus \alpha(d_n)$. Hence, the probability that $\alpha(\mathcal{P}_v) = c$ is the sum of all the products $\mathsf{Pr}(\alpha(\mathcal{P}_1) = c_1) \cdots \mathsf{Pr}(\alpha(\mathcal{P}_n) = c_n)$ such that $c = c_1 \oplus \cdots \oplus c_n$. Therefore, the distribution $\alpha(\widehat{\mathcal{P}}_v)$ is the convolution of the distributions $\alpha(\widehat{\mathcal{P}}_i)$ with respect to $\oplus$, that is,

$$\alpha(\widehat{\mathcal{P}}_v) = \alpha(\widehat{\mathcal{P}}_1) *_\oplus \cdots *_\oplus \alpha(\widehat{\mathcal{P}}_n), \tag{3}$$

where the convolution with respect to $\oplus$ is defined as $(f *_\oplus g)(c) = \sum_{c_1 \oplus c_2 = c} f(c_1) \cdot g(c_2)$ for any two functions $f, g \colon M \to \mathbb{R}$.

For *det* nodes v, the same equation applies, although the supporting arguments are a bit more complicated. The crucial difference is that for *det* nodes, $[\![\widehat{\mathcal{P}_v}]\!]$ is a space of forests, not trees, since the trees (or forests) in the $[\![\widehat{\mathcal{P}_i}]\!]$ are combined without attaching them to a new root. For a fully formalized argument, one would have to generalize the syntax to trees with distributional roots and the semantics to one where p-documents are interpreted by px-spaces over labeled forests, which is tedious, but not difficult.

We summarize how one can use the operations introduced to obtain the distribution of a monoid aggregate function over a *mux-det* document.

THEOREM 7.1. *Let α be a monoid aggregation function and $\widehat{\mathcal{P}} \in \mathsf{PrXML}^{mux,det}$. Then $\alpha(\widehat{\mathcal{P}})$ can be obtained in a bottom-up fashion by*

(1) *attaching a Dirac distribution to every leaf and for every occurrence of the empty document;*
(2) *taking convex sums at every mux node; and*
(3) *taking convolutions with respect to α at each det and each regular non-leaf node.*

We now illustrate how to apply Theorem 7.1 on an example.

Example 7.2. We compute the distribution $Q_{\mathrm{Bonus}}^{\min}(\widehat{\mathcal{P}}_{\mathrm{PER\text{-}L}})$. Since $Q_{\mathrm{Bonus}}^{\min}$ is an $\mathsf{SP}^{\min}$ query, due to Lemma 5.5, we can compute a p-subdocument of $\widehat{\mathcal{P}}_{\mathrm{PER\text{-}L}}$, denoted $\widehat{\mathcal{P}}_{\mathrm{BON\text{-}L}}$, such that $Q_{\mathrm{Bonus}}^{\min}(\widehat{\mathcal{P}}_{\mathrm{PER\text{-}L}}) = \min(\widehat{\mathcal{P}}_{\mathrm{BON\text{-}L}})$. The document $\widehat{\mathcal{P}}_{\mathrm{BON\text{-}L}}$ can be obtained from $\widehat{\mathcal{P}}_{\mathrm{PER\text{-}L}}$ by removing all the leaves that are not bonuses and the paths leading to them from the root, see Figure 6, left. Since we are computing a distribution of min, the operation $\oplus$ is min itself, that we will use in infix notation. Using Theorem 7.1 we obtain that the distribution $\min(\widehat{\mathcal{P}}_{\mathrm{BON\text{-}L}})$ is

$$((0.1\,\delta_{25} + 0.9\,(\delta_{44}\,\mathsf{min}\,\delta_{50}))\,\mathsf{min}\,\delta_{50})\,\mathsf{min}\,(0.7\,(\delta_{15}\,\mathsf{min}\,\delta_{44}) + 0.3\,\delta_{15}).$$

Since for every $a \leqslant b$ we have $(\delta_a\,\mathsf{min}\,\delta_b) = \delta_a$ the expression for $\min(\widehat{\mathcal{P}}_{\mathrm{BON\text{-}L}})$ can be simplified to $\big((0.1\,\delta_{25} + 0.9\,\delta_{44})\,\mathsf{min}\,\delta_{50}\big)\,\mathsf{min}\,\delta_{15}$. Moreover, since $25 < 50$ and $44 < 50$ we can further simplify the expression to $(0.1\,\delta_{25} + 0.9\,\delta_{44})\,\mathsf{min}\,\delta_{15}$. This expression represents the distribution $\min(\widehat{\mathcal{P}}_{\mathrm{BON\text{-}L}})$.

Since for any $\widehat{\mathcal{P}}$ the carrier of $\min(\widehat{\mathcal{P}})$ and of $\mathsf{count}(\widehat{\mathcal{P}})$ has at most as many elements as there are leaves in $\widehat{\mathcal{P}}$, we can draw some immediate conclusions from Theorem 7.1.

COROLLARY 7.3. *For any $\widehat{\mathcal{P}} \in \mathsf{PrXML}^{mux,det}$,*

(1) *the distributions $\mathsf{count}(\widehat{\mathcal{P}})$, $\min(\widehat{\mathcal{P}})$ can be computed in time polynomial in $|\widehat{\mathcal{P}}|$;*
(2) *the distribution $\mathsf{sum}(\widehat{\mathcal{P}})$ can be computed in time polynomial in $|\widehat{\mathcal{P}}| + |\mathsf{sum}(\widehat{\mathcal{P}})|$.*

PROOF SKETCH. Claim 1 holds because computing a convex sum and convolutions with respect to "+" and min of two distributions is polynomial and all distributions involved in computing $\mathsf{count}(\widehat{\mathcal{P}})$ and $\min(\widehat{\mathcal{P}})$ have size $O(|\widehat{\mathcal{P}}|)$. Claim 2 holds because, in addition, a convex sum and the convolution with respect to "+" of two distributions have at least the size of the largest of the two arguments. □

Remark. Eq. (3) is in fact a special case of a general principle: If X and Y are two M-valued random variables on the probability spaces $\mathcal{X}$, $\mathcal{Y}$, with distributions f, g, respectively, then the distribution of $X \oplus Y \colon \mathcal{X} \times \mathcal{Y} \to M$ is the convolution $f *_M g$ of f and g. This principle has also been applied by Ré and Suciu [2007] for queries with aggregation constraints over probabilistic relational databases.

7.2. Monoid Aggregates over Continuous Probabilistic Data

We now discuss how the results for single-path aggregate queries with monoid functions obtained in the discrete case can be lifted to the continuous case $\mathsf{PrXML}_{cont}^{mux,det}$.

As discussed in Section 4, one can compute distributions $Q^\alpha(\widehat{\mathcal{P}})$ of aggregate queries by first computing (possibly exponentially many) skeletons of $\widehat{\mathcal{P}}$, evaluating Q^α over the skeletons, which can be done using convolutions, and then combining the resulting distributions with convex sums. It turns out that the computation of the skeletons is not needed for *mux-det* p-documents. One can apply the same bottom-up evaluation technique for SP^α queries with monoid functions as discussed in Theorem 7.1, with the difference that one skips step 1, because the distributions are already attached to the leaves of continuous p-documents.

Obviously, there is no hope of computing probabilities of aggregate query answers if it is not possible to somehow combine (either symbolically or numerically) the probability distributions of the leaves. Proposition 4.2 hints that if we are able to efficiently apply a number of basic operations on our probability distribution functions, we are able to compute the distribution of the min, max, or sum. The following operations are required: convex sums (for *mux* nodes); convolution (for sum, in conjunction with *det* nodes); integration and multiplication (for min and max, in conjunction with *det* nodes).

One simple case where we can perform these operations efficiently is when *cont* leaves are piecewise polynomials of a bounded degree, that is, for $\mathsf{PP}(K)$. It is reasonable to assume that such a bound K exists for every application. This bound ensures that the piecewise polynomial representing the distribution of the query answer has degree polynomial in the size of the document. Formally, we have the following.

PROPOSITION 7.4. *For a p-document in* $\mathsf{PrXML}^{mux,det}$ *with l leaves, where the leaves are labeled with distributions in* $\mathsf{PP}(K)$*, the polynomials in the distributions of* max *and* min *have degree at most* $K \cdot l$.

This proposition combined with Proposition 4.2 gives the following result.

THEOREM 7.5. *For p-documents in* $\mathsf{PrXML}_{cont}^{mux,det}$ *that are labeled with distributions in* $\mathsf{PP}(K)$*, for a fixed natural number K, we have:*

(1) *The distribution of results of queries in* $\mathsf{SP}^{\mathsf{sum}}$ *can be computed in polynomial time in the combined size of the input and the output.*
(2) *The distribution of results of queries in* $\mathsf{SP}^{\mathsf{max}}$ *and* $\mathsf{SP}^{\mathsf{min}}$ *can be computed in polynomial time.*
(3) *All moments of results of queries in* $\mathsf{SP}^{\mathsf{sum}}$*,* $\mathsf{SP}^{\mathsf{max}}$ *and* $\mathsf{SP}^{\mathsf{min}}$ *can be computed in polynomial time.*

PROOF. Proofs for both Claims 1 and 2 are based the following observation. During the convolution of pdfs, the size of the larger one never decreases. This holds clearly in the discrete case. It also works for the continuous case, if, for example, pdfs are $\mathsf{PP}(K)$ functions, for a fixed K.

Indeed, suppose f is a pdf in $\mathsf{PP}(K)$, with interval endpoints $x_0, \ldots, x_n$, where the restriction to the interval (x_{i-1}, x_i) is a polynomial f_i. If we convolve f with a polynomial g, this amounts to multiplying (and integrating) f and g. Multiplying f_i, f_{i+1} by another function cannot make the results $f_i \cdot g$ and $f_{i+1} \cdot g$ equal if f_i, f_{i+1} were not equal, and integration cannot make them equal either. Hence the size of the larger pdf never decreases.

Observe that for $Q \in \mathsf{SP}$ the resulting distributions $Q^{\mathsf{min}}(\widehat{\mathcal{P}})$ and $Q^{\mathsf{max}}(\widehat{\mathcal{P}})$ for a $\widehat{\mathcal{P}}$ labeled with distributions from $\mathsf{PP}(K)$ can be computed (due to Proposition 5.5) on a p-subdocument $\widehat{\mathcal{P}}_Q$ of $\widehat{\mathcal{P}}$ as $\mathsf{min}(\widehat{\mathcal{P}}_Q)$ and $\mathsf{max}(\widehat{\mathcal{P}}_Q)$ and, therefore, are limited by $\widehat{\mathcal{P}}$,

Table IV. Data Complexity of Query Evaluation over PrXML^{mux,det}

PrXML^{mux,det}	Aggregate query language			
	SP, TP		TPJ	
Membership	sum, avg, countd count, min	NP-complete in P	sum, avg, countd count, min	NP-complete in NP
Probability	sum*, avg, countd count, min	FP$^{\#P}$-complete in P	FP$^{\#P}$-complete	
	SP	TP		
Moments	in P	avg in FP$^{\#P}$ others in P	FP$^{\#P}$-complete	

*SP^{sum} query evaluation is in P when the complexity is measured in the size of the input plus the resulting distribution.

while $\mathsf{sum}(\widehat{\mathcal{P}}_Q)$ and consequently $Q^{\mathsf{sum}}(\widehat{\mathcal{P}})$ can be of exponential size (see Theorem 8.16, Claim 2). Combining this with the observation that convolution never decreases, we conclude the proof of Claims 1 and 2.

Claim 3 for SP$^{\mathsf{max}}$ and SP$^{\mathsf{min}}$ follows from Claim 2 and the fact that moments of any degree can be computed from the distributions in polynomial time. Claim 3 for SP$^{\mathsf{sum}}$ follows from Theorem 6.10. □

Another class of distributions for which we can compute (representations of) distributions for sum is the class of convex sums of Gaussian distributions. It is well known [Ash and Doléans-Dade 2000] that Gaussians are closed under convolution $*$ for sum, that is, $N(\mu_1, \sigma_1^2) * N(\mu_2, \sigma_2^2) = N(\mu_1 + \mu_2, \sigma_1^2 + \sigma_2^2)$ and hence convex sums of Gaussians are closed under convex sum and convolution. Thus, we obtain the following result.

THEOREM 7.6. *For p-documents in* PrXML$^{mux,det}_{cont}$ *that are labeled with convex sums of Gaussian distributions, the distribution of results of* SP$^{\mathsf{sum}}$ *queries is a convex sum of Gaussian distributions and can be computed in polynomial time in the combined size of the input and the output.*

We now illustrate how to apply Theorem 7.6 on an example.

Example 7.7. For $\widehat{\mathcal{P}}_{\mathrm{CONS}}$ (Figure 5) we want to know the total heat consumption, which can be computed by the following query $Q^{\mathsf{sum}} \in$ SP$^{\mathsf{sum}}$ expressed in XPath: sum(//value/text()). Using Lemma 5.5, we can compute a p-subdocument $\widehat{\mathcal{P}}_{\mathrm{MEAS}}$ of $\widehat{\mathcal{P}}_{\mathrm{CONS}}$, that is presented on Figure 6 (right), such that $Q^{\mathsf{sum}}(\widehat{\mathcal{P}}_{\mathrm{CONS}}) = \mathsf{sum}(\widehat{\mathcal{P}}_{\mathrm{MEAS}})$. Now using Theorem 7.1 we obtain the distribution $\mathsf{sum}(\widehat{\mathcal{P}}_{\mathrm{MEAS}})$.

$$N(15, 3) * (0.1\, N(50, 5) + 0.9\, N(52, 5)) * N(200, 8)$$

We finally distribute the convolutions with $N(15, 3)$ and $N(200, 8)$ over the convex sum. Since $N(15, 3) * N(50, 5) * N(200, 8) = N(265, 16)$ and $N(15, 3) * N(52, 5) * N(200, 8) = N(267, 16)$, we obtain $Q^{\mathsf{sum}}(\widehat{\mathcal{P}}_{\mathrm{CONS}}) = 0.1\, N(265, 16) + 0.9\, N(267, 16)$.

Other results from the discrete case can be generalized to the continuous case, too. For example, it can be shown that moments of queries in TP$^{\mathsf{sum}}$ can be computed in polynomial time over PrXML$^{mux,det}_{cont}$ (and similarly for SP$^{\mathsf{sum}}$ and PrXML$^{cie}_{cont}$), by replacing the *cont* nodes by the expected value of the represented distribution.

8. AGGREGATING P-DOCUMENTS WITH LOCAL DISTRIBUTIONAL NODES

We investigate the three computational problems for aggregate queries for the restricted class of PrXML^{mux,det}, drawing upon the principles developed in the preceding section. Table IV gives an overview of the data complexity results obtained in this section.
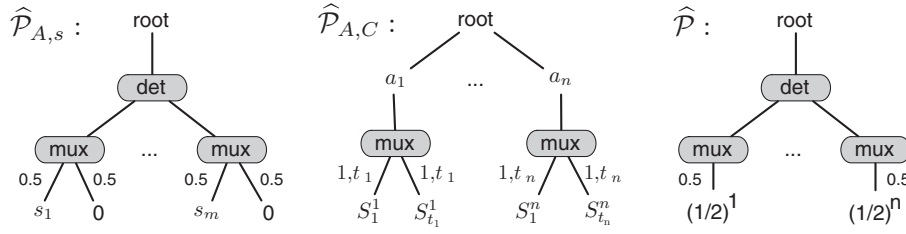
Fig. 10. Left: p-document $\widehat{\mathcal{P}}_{A,s}$ for reduction of the Subset-Sum problem to the membership problem for sum and avg (Lemma 8.2). Center: p-document $\widehat{\mathcal{P}}_{A,C}$ for reduction of the Exact-Cover-By-3-Sets problem to the membership problem for countd (Lemma 8.3). Right: p-document for exhibiting an exponential explosion of sum($\widehat{\mathcal{P}}$) distribution for $\widehat{\mathcal{P}} \in \mathsf{PrXML}^{mux,det}$ (Lemma 8.6).

8.1. Membership

THEOREM 8.1. *Let* $\alpha \in$ {sum, count, min, avg, countd}. *Then membership over* $\mathsf{PrXML}^{mux,det}$ *is in* NP *for the class* TPJ^{α}. *Moreover, the problem is*

(1) NP-*hard for every query in* SP^{sum}, SP^{avg}, SP^{countd};
(2) *of polynomial combined complexity for the classes* SP^{min} *and* SP^{count};
(3) *of polynomial data complexity for any query in* TP^{min} *and* TP^{count}.

PROOF. The NP upper bound is inherited from the *cie* case (Theorem 6.2). Claims 2 and 3 follow from their counterparts (Claims 1 and 2, respectively) in Theorem 8.7 further. Claim 1 can be shown by a reduction of Subset-Sum and Exact-Cover-By-3-Sets, as presented in the following Lemmas 8.2 and 8.3, respectively. □

The next lemma highlights why membership is difficult for sum and avg. The Subset-Sum problem [Garey and Johnson 1979] is, given a finite set A, an $\mathbb{N}$-valued function s on A, and $c \in \mathbb{N}$, to decide whether there is some $A' \subseteq A$ such that $\sum_{a \in A'} s(a) = c$.

LEMMA 8.2. *For every set A, $\mathbb{N}$-valued (weight) function s on A, and $c \in \mathbb{N}$ one can compute in time polynomial in $|A| + \sum_{a \in A} |s(a)|$ a p-document $\widehat{\mathcal{P}}_{A,s} \in \mathsf{PrXML}^{mux,det}$ such that the following are equivalent:*

(1) *there is $A' \subseteq A$ such that $\sum_{a \in A'} s(a) = c$,*
(2) $\mathsf{Pr}(\mathsf{sum}(\mathcal{P}_{A,s}) = c) > 0$,
(3) $\mathsf{Pr}(\mathsf{avg}(\mathcal{P}_{A,s}) = c \cdot |A|) > 0$.

PROOF. Let $A := \{a_1, \ldots, a_m\}$. Then $\widehat{\mathcal{P}}_{A,s}$ (Figure 10, left) has one *det* child under the root, which has m children $v_1, \ldots, v_m$ that are *mux* nodes. Each v_i has two children: a_i labeled with $s_i := s(a_i)$ and b_i labeled with 0. The edges to both a_i and b_i are labeled with 0.5. Clearly, $\widehat{\mathcal{P}}_{A,s}$ can be constructed in polynomial time in the size of A. Observe that c is a possible value for sum if and only if in some world the only children of the root are a_i's with labels summing-up to c, that is, if and only if there is $A' \subseteq A$ consisting of these a_i's with the weights summing-up to c.

Furthermore, in every world of the p-document $\widehat{\mathcal{P}}_{A,s}$ the number of leaves is the same and equal to m. Therefore, c is a possible value for sum if and only if $c \cdot m$ is a possible value for avg. □

The next lemma highlights why membership is difficult for countd. Recall that Exact-Cover-By-3-Sets is, given a finite set A, such that $|A| = 3q$ and a collection C of 3-element subsets of A, to decide whether there is some $C' \subseteq C$ such that every element of A occurs in exactly one member of C'. This problem is NP-hard [Garey and Johnson 1979].

LEMMA 8.3. *For every set A, where $|A| = 3q$ and every collection C of 3-element subsets of A, one can compute in polynomial time a p-document $\widehat{\mathcal{P}}_{A,C} \in \mathsf{PrXML}^{mux,det}$ such that the following are equivalent:*

(1) *there is $C' \subseteq C$ such that every $a \in A$ occurs in exactly one member of C';*
(2) $\mathsf{Pr}(\mathsf{countd}(\mathcal{P}_{A,C}) = q) > 0$.

PROOF. Let $A = \{a_1, \ldots, a_n\}$, and $C = \{S_1, \ldots, S_m\}$. Assume for each i that $S_1^i, \ldots, S_{t_i}^i$ are all the members of C that contain a_i. Then $\widehat{\mathcal{P}}_{A,C}$ (Figure 10, center) has n children $a_1, \ldots, a_n$ below the root. Each a_i has exactly one child that is a *mux* node with t_i children, namely $S_1^i, \ldots, S_{t_i}^i$. Each S_j^i is a leaf and the edge to it is labeled with $1/t_i$. Intuitively, each node S_j^i indicates that an element a_i in A is "covered" with a set S_j^i in C'.

The probability $\mathsf{Pr}(\mathsf{countd}(\mathcal{P}_{A,C}) = q)$ is more than zero if and only if there is a world in $[\![\widehat{\mathcal{P}}_{A,C}]\!]$ such that it has exactly q distinct leaves S_j^i, namely, $S_{j_1}^{i_1}, \ldots, S_{j_q}^{i_q}$, if and only if, the set $C' = \{S_{j_1}^{i_1}, \ldots, S_{j_q}^{i_q}\}$ covers A and, since the cardinality of C' is q, every element of A occurs in exactly one member of C'. □

8.2. Probability Computation

We start with the case of sum queries for which probability computation over $\mathsf{PrXML}^{mux,det}$ is tractable in terms of the size of the output distribution.

THEOREM 8.4. *Probability computation over $\mathsf{PrXML}^{mux,det}$ is in $\mathsf{FP}^{\#P}$ for the class TPJ^{sum}. Moreover,*

(1) *the problem is polynomial in the size of the input and overall distribution for every $Q \in \mathsf{SP}^{sum}$;*
(2) *the problem is $\mathsf{FP}^{\#P}$-complete in the size of the input p-document for SP^{sum};*
(3) *there is a p-document for which the distribution of every query in SP^{sum} is exponentially large in the size of the p-document.*

PROOF. The upper bound is inherited from the *cie* case (Theorem 6.2). Claim 1 follows from Corollary 7.3. Finally, Proposition 5.6 together with the following Lemma 8.5 yields Claim 2 and together with Lemma 8.6 it yields Claim 3. □

We now show why probability computation is difficult for sum. The #Subset-Sum problem is the counting counterpart of Subset-Sum, and *counts* the number of all $A' \subseteq A$ such that $\sum_{a \in A'} s(a) = c$; #Subset-Sum is a #P-hard problem.

LEMMA 8.5. *For every set A, $\mathbb{N}$-valued (weight) function s on A, and $c \in \mathbb{N}$ one can compute in polynomial time a p-document $\widehat{\mathcal{P}}_{A,s} \in \mathsf{PrXML}^{mux,det}$ and a number $c' \in \mathbb{Q}$ such that the following are equivalent:*

(1) *the number of $A' \subseteq A$ such that $\sum_{a \in A'} s(a) = c$ is k,*
(2) *the probability $\mathsf{Pr}(\mathsf{sum}(\mathcal{P}_{A,s}) = c)$ is $k \cdot c'$.*

PROOF. Consider the document $\widehat{\mathcal{P}}_{A,s}$ from 8.2 (Figure 10, left) and let $c' = 1/2^m$. Then

$$\mathsf{Pr}(\mathsf{sum}(\mathcal{P}_{A,s}) = c) = \sum_{\mathcal{P} \in W} \mathsf{Pr}(\mathcal{P}) = |W|/2^m = k/2^m = k \cdot c',$$

where W is the set of all documents $\mathcal{P}$ in $[\![\widehat{\mathcal{P}}_{A,s}]\!]$ such that $\mathsf{sum}(\mathcal{P}) = c$. □

We now see a p-document with an exponentially large distribution of sum.

LEMMA 8.6.   *There exists a p-document $\widehat{\mathcal{P}}$ in* PrXML^{mux,det} *such that the distribution of* sum$(\mathcal{P})$ *is exponential in* $|\widehat{\mathcal{P}}|$, *that is,* $|\mathsf{sum}(\widehat{\mathcal{P}})| = 2^{|\widehat{\mathcal{P}}|}$.

PROOF.   Consider $\widehat{\mathcal{P}}$ (Figure 10, right), that has a root with one *det* child that in turn has *n mux* children the i-th of which has one child v_i labeled with the number $(1/2)^i$, and the edges to v_i are labeled with 0.5. Then it is easy to see that $|\mathsf{sum}(\widehat{\mathcal{P}})| = 2^n$, where n is the number of leaves in $\widehat{\mathcal{P}}$.   □

We conclude with probability computation for count, min, avg, and countd.

THEOREM 8.7.   *Let* $\alpha \in \{\mathsf{count, min, avg, countd}\}$. *Then probability computation over* PrXML^{mux,det} *is in* FP$^{\#P}$ *for the class* TPJ$^{\alpha}$. *Moreover, the problem is*

(1) *of polynomial combined complexity for the classes* SP$^{\mathsf{min}}$ *and* SP$^{\mathsf{count}}$;
(2) *of polynomial data complexity for any query in* TP$^{\mathsf{min}}$ *and* TP$^{\mathsf{count}}$;
(3) *#P-hard for any query in* SP$^{\mathsf{avg}}$ *and* SP$^{\mathsf{countd}}$;
(4) *#P-hard for some query in* TPJ$^{\mathsf{count}}$ *and* TPJ$^{\mathsf{min}}$.

PROOF.   The FP$^{\#P}$ upper bound is inherited from the *cie* case (Theorem 6.2). Claim 1 follows from Corollary 7.3, since, due to Proposition 5.5, for an aggregate SP query Q^{α} we have that $Q^{\alpha}(\widehat{\mathcal{P}}) = \alpha(\widehat{\mathcal{P}}_Q)$.

Regarding Claim 2, algorithms for count and min can be developed in a straightforward way, applying the techniques from Cohen et al. [2008] to evaluate TP-queries with aggregate constraints. For a given p-document, there are only linearly many possible values for min and count, the probability of which can be computed in polynomial time by incorporating them in constraints. Consequently, the entire distribution of min or count can be computed in polynomial time.

Claim 3 for countd can be shown by a reduction of the #K-cover problem, that is known to be #P-complete [Ré and Suciu 2007]. Claim 3 for avg can be shown by a straightforward reduction from the #P-complete problem #Non-Negative-Subset-Average introduced in Ré and Suciu [2007].[7]

Claim 4 follows from Corollary 5.9.   □

8.3. Moment Computation

THEOREM 8.8.   *Let* $\alpha \in \{\mathsf{sum, count, min, avg, countd}\}$. *Then computation of moments of any degree over* PrXML^{mux,det} *is in* FP$^{\#P}$ *for the class* TPJ$^{\alpha}$. *Moreover, the problem is*

(1) *of polynomial combined complexity for the class* SP$^{\alpha}$;
(2) *of polynomial data complexity for the class* TP$^{\alpha}$, *if* $\alpha \neq \mathsf{avg}$;
(3) *#P-hard for some query in* TPJ$^{\alpha}$.

PROOF.   The FP$^{\#P}$ upper bound is inherited from the *cie* case (Theorem 6.3).

Claim 3 follows from Corollary 5.9. Proofs of Claim 1 and Claim 2 will be presented after the following supporting lemmas.   □

The next lemma shows that computation of an auxiliary probability, that will be used in the proof of Claim 1 of the preceding theorem, is tractable.

LEMMA 8.9.   *There is a polynomial-time algorithm that computes, given a p-document* $\widehat{\mathcal{P}} \in$ PrXML^{mux,det} *and leaves* $v_1, \ldots, v_k$ *occurring in* $\widehat{\mathcal{P}}$, *the probability*

$$\mathsf{Pr}(\{d \in [\![\widehat{\mathcal{P}}]\!] \mid v_1, \ldots, v_k \text{ occur in } d\}). \tag{4}$$

---

[7]The same problems has been used earlier by Ré and Suciu [2007] to show #P-hardness of evaluating relational queries with countd and avg HAVING constraints.

One way to see the algorithm is to convert the $\mathsf{PrXML}^{mux,det}$ p-document into a PrXML^{cie} p-document, then gather all variables from the root to each leaf in a single conjunction, and compute the probability of this conjunction.

We show that computation of moments of any degree for countd over a px-space, regardless whether it can be represented by a p-document, can be polynomially reduced to computation of an auxiliary probability.

LEMMA 8.10. *Let $\mathcal{S}$ be a* px-*space. For every document $d \in \mathcal{S}$, let $L(d)$ be the set of all leaf labels of d and $L = \bigcup_{d \in \mathcal{S}} L(d)$. Let* $\mathsf{countd}_{\mathcal{S}}$ *be the random variable defined by* countd *on $\mathcal{S}$. Then*

$$\mathbb{E}\big(\mathsf{countd}_{\mathcal{S}}{}^k\big) = \sum_{(l_1,\dots,l_k) \in L^k} \mathsf{Pr}\left(\{d \in \mathcal{S} \mid l_1, \dots, l_k \text{ occur in } L(d)\}\right),$$

where the last term denotes the probability that the set of leaf labels of a random document $d \in \mathcal{S}$ contains all distinct labels $l_1, \dots, l_k$.

The proof is similar to the one of regrouping sums for sum; see Lemma 6.5.

The auxiliary probability introduced in the previous lemma can be in fact computed in polynomial time for px-spaces represented by $\widehat{\mathcal{P}} \in \mathsf{PrXML}^{mux,det}$.

PROPOSITION 8.11. *There is a polynomial-time algorithm that computes, given $\widehat{\mathcal{P}} \in$ $\mathsf{PrXML}^{mux,det}$ and leaf labels $l_1, \dots, l_k$ occurring in $\widehat{\mathcal{P}}$, the probability*

$$\mathsf{Pr}\left(\{d \in [\![\widehat{\mathcal{P}}]\!] \mid l_1, \dots, l_k \text{ occur in } L(d)\}\right).$$

PROOF. We reduce computation of this probability to evaluation of a TP query over $\mathsf{PrXML}^{mux,det}$ p-documents. Since the latter evaluation is polynomial [Kimelfeld et al. 2009], the probability computation is polynomial as well. Following XPath notation, the TP query that we use to compute the probability is the following:
`//text()="l_1" and ... and //text()="l_k"`. □

Polynomiality of moment computation for avg is more involved. In order to show it we present three lemmas. The first lemma shows that for every p-document there is an equivalent p-document where each *mux* and *det* node has at most two children and this equivalent p-document can be computed in polynomial time. The second lemma shows that moment computation for avg is reducible to conditional moment computation for sum. The third lemma shows that conditional moments computation for sum is polynomial for p-documents with bounded branching of *det* nodes.

LEMMA 8.12. *Let $\widehat{\mathcal{P}}$ be a p-document in $\mathsf{PrXML}^{mux,det}$. Then one can compute in polynomial time a p-document $\widehat{\mathcal{Q}} \in \mathsf{PrXML}^{mux,det}$ such that every det and mux node of $\widehat{\mathcal{Q}}$ has at most two children and $[\![\widehat{\mathcal{P}}]\!] = [\![\widehat{\mathcal{Q}}]\!]$.*

Consider now an analog of the Regrouping Sums Lemma (6.5) for avg.

LEMMA 8.13. *Let $\widehat{\mathcal{P}} \in \mathsf{PrXML}^{mux,det}$ be a p-document. Then*

$$\mathbb{E}(\mathsf{avg}(\mathcal{P})^n) = \sum_{j \in \mathbb{N}^+} \left(\frac{1}{j^n} \times \mathsf{Pr}(\mathsf{count}(\mathcal{P}) = j) \times \mathbb{E}(\mathsf{sum}(\mathcal{P})^n \mid \mathsf{count}(\mathcal{P}) = j) \right).$$

The next lemma shows that the conditional moments for sum occurring in the previous lemma can be computed in polynomial time.

LEMMA 8.14. *There is an algorithm that computes the conditional expected value $\mathbb{E}(\mathsf{sum}(\mathcal{P})^n \mid \mathsf{count}(\mathcal{P}) = j)$ in time polynomial in j and linear in $|\widehat{\mathcal{P}}|$.*

Now we are ready to complete the proof of Theorem 8.8.

PROOF OF THEOREM 8.8, CLAIM 1 AND CLAIM 2.

Claim 1. All our algorithms for computing moments for SP^α first reduce aggregate query answering to function evaluation (see Proposition 5.5). Now we show polynomiality of function evaluation for all the five functions.

Case of count, sum: The algorithm for count and sum is a refinement of the one for the *cie* case (Theorem 6.3) and follows from Lemma 6.5 and Lemma 8.9.

Case of min: The algorithm for min computes the moments exploiting the entire distribution $\min(\widehat{\mathcal{P}})$, which can be computed in polynomial time (Corollary 7.3).

Case of countd: Intuitively, for countd we apply similar techniques of regrouping sums to those that we used for sum in Lemma 6.5. In doing so, we exploit the fact that the probability for a value (or sets of values of fixed cardinality) to occur in a query result over a *mux-det* p-document can be computed in polynomial time, which follows from Kimelfeld et al. [2008]. More precisely, by Lemma 8.10, the k-th moment of countd over $\widehat{\mathcal{P}}$ is the sum of $|L|^k$ elements, where L is the set of leaf labels in $\widehat{\mathcal{P}}$. Each element in the sum can be computed in polynomial time by Proposition 8.11.

Case of avg: Intuitively, the algorithm for avg traverses p-documents in a bottom-up fashion. It maintains conditional moments of sum for each possible value of count and combines them in two possible ways, according to the node types.[8]

More precisely, by Lemma 8.13, the k-th moment of avg over $\widehat{\mathcal{P}}$ is an infinite sum of products. In fact only finitely many of these products are nonzero. Let t be the number of the leaves in $\widehat{\mathcal{P}}$, then the second component of the product $\Pr(\text{count}(\mathcal{P}) = j)$ is zero if $j > t$. Hence, there are at most t, that is, linearly many nonzero products. What about the components of the products? The first component $1/j^n$ can be computed in polynomial time since j is bounded by the size of $\widehat{\mathcal{P}}$. The second component of the product $\Pr(\text{count}(\mathcal{P}) = j)$ can be computed in polynomial time due to Theorem 8.7. The third component of the product $\mathbb{E}(\text{sum}(\mathcal{P})^n \mid \text{count}(\mathcal{P}) = j)$ can be computed in polynomial time due to Lemma 8.14 and the fact the $j \leqslant |\widehat{\mathcal{P}}|$. Therefore, the k-th moment for avg can be computed in time polynomial in $|\widehat{\mathcal{P}}|$.

Claim 2. Moments for count and min can be computed from the distributions, which can be constructed in polynomial time as in the proof of Theorem 8.7 (2).

Algorithms for sum and countd can be based on a generalization of the principle of regrouping sums (see Lemma 6.5) for tree-pattern queries.

Analogously as for the case of single-path queries, the crucial element for the complexity of the sum algorithm is the difficulty of computing the probability that a node (or sets of nodes of fixed cardinality) occurs in a query result. For tree-pattern queries without joins, these probabilities can be computed in polynomial time adapting the techniques from Kimelfeld et al. [2008]. A variation of this principle, where the probability of a given set of values to occur in a query result is computed, gives an algorithm for countd. □

8.4. Aggregating Continuous PrXML^{mux,det}

We now investigate under which conditions the results for discrete p-documents PrXML^{cie} extend to the continuous case. Table V gives an overview of the results that are proved next.

Membership. We start with hardness of membership for sum and avg.

THEOREM 8.15. *The membership problem is* NP-*hard for every query in* SP^{sum} *and* SP^{avg} *over p-documents labeled with* PP(0) *distributions.*

------

[8]A technique that is similar in spirit has been presented by Jayram et al. [2007] for probabilistic streams.

Table V. Data Complexity of Query Evaluation over $\mathsf{PrXML}_{cont}^{mux,det}$

$\mathsf{PrXML}_{cont}^{mux,det}$	Aggregate query language			
	SP		TP	
Membership	sum, avg	NP-complete*	sum, avg	NP-complete*
	min	in P†	min	in NP*
Probability	sum‡, avg	FP$^{\#P}$-complete†	sum, avg	FP$^{\#P}$-complete†
	min	in P†	min	in FP$^{\#P\dagger}$
Moments	min, sum	in P†	in FP$^{\#P\dagger}$	
	avg	in FP$^{\#P\dagger}$		

*NP membership holds for min-reasonable (for TP^{min}) and sum-reasonable (for TP^{sum} and TP^{avg}) p-documents.
†The upper bound holds for p-documents labeled with distributions in PP(K), for a fixed K.
‡SP^{sum} query evaluation over p-documents with distributions in PP (K), for a fixed K, is in P when the complexity is measured in the size of the input plus the resulting distribution.
In all cases the hardness already holds for p-documents labeled with PP(0) distributions.

PROOF SKETCH. NP-hardness for sum and avg can be shown by first applying Proposition 5.6 and then using the same reduction as for the discrete case (see Lemma 8.2) and narrow PP(0) characteristic functions as discussed in Lemma 6.8. More precisely, the proof of Lemma 8.2 should be modified by taking the probability $\mathsf{Pr}(\mathsf{sum}(\mathcal{P}_{A,s}) \in [c-1/2, c+1/2])$, while the a_i leaves of $\widehat{\mathcal{P}}_{A,s}$ should be labeled with $[s_i - \varepsilon, s_i + \varepsilon]$, for $\varepsilon < 1/2^m$, where m is the cardinality $|A|$. □

We now summarize complexity results for different aggregate functions and distributions labeling p-documents that directly follow from previous results.

Membership is in NP for the class (1) TP^{min} over min-reasonable $\mathsf{PrXML}_{cont}^{mux,det}$ p-documents; (2) TP^{sum} over TP^{avg} and sum-reasonable $\mathsf{PrXML}_{cont}^{mux,det}$ p-documents. Moreover, the problem is (3) of polynomial data complexity for the class SP^{min} over p-documents labeled with distributions in PP(K); (4) of polynomial complexity in the size of both input document and output distribution for the class SP^{sum} over p-documents labeled with distributions in PP(K), for a fixed K. Claims 1 and 2 follow from Theorem 6.7 and Claims 3 and 4 follow from Theorem 7.5.

Probability Computation. Let us now look at probability computation for sum.

THEOREM 8.16. *Probability computation over* $\mathsf{PrXML}_{cont}^{mux,det}$ *is in* FP$^{\#P}$ *for the class* TP^{sum}. *Moreover,*

(1) *the problem is* FP$^{\#P}$*-complete in the size of the input p-document for the class* SP^{sum} *over p-documents labeled with distributions from* PP(0)*;*
(2) *there is a p-document labeled with distributions from* PP(0) *for which the support of the distribution (with respect to the support of the original p-document) of every query in* SP^{sum} *has exponentially many distinct and not bordering intervals.*

PROOF. Both claims are continuous counterparts of Theorem 8.4, and the hardness can be proved analogously to the hardness for the discrete case, using narrow PP(0) characteristic functions on the leaves as discussed in Lemma 6.8.

Claim 1 can be proved using a reduction from #Subset-Sum as in Lemma 8.5 extended with PP(0) functions as in the proof of hardness for Theorem 8.15.

Claim 2 can be proved using the p-document from the proof of Lemma 8.6 extended with continuous labels from PP(0). The support of each distribution labeling the leaves is defined using a similar construction as in Cantor's diagonal argument. Each leaf i is

labeled with a function that has the support $[a_i, b_i]$ where a_i and b_i are numbers with $2n + 1$ digits, starting with 1 and with $2n - 1$ zeros on all remaining $2n$ digits, but the $2i$-th and $(2i + 1)$-th, respectively. The distribution of sum over such a document is the convolution of all the distributions on the leaves and one can easily see that its support is the union of (exponentially many in *n*) disjoint intervals obtained by all possible combinations of the borders of each $[a_i, b_i]$.

$\mathsf{FP}^{\#\mathsf{P}}$-membership for Claim 1 is inherited form the the $\mathsf{PrXML}^{cie}_{cont}$ case. $\square$

A corollary of Theorem 8.16 is that probability computation is #P-hard for any query in $\mathsf{SP}^{\mathsf{avg}}$ and p-documents labeled with distributions from PP(0). A proof uses the fact that the number of leaves in each world of a skeleton is the same as in the skeleton.

Summing up complexity for different aggregate functions and distributions labeling p-documents, we have that, due to Theorem 7.5, probability computation is polynomial in the size of the input and the overall distribution for every query in $\mathsf{SP}^{\mathsf{sum}}$ over p-documents labeled with distributions from PP(*K*), for a fixed *K*. An upper bound on probability computation for $\mathsf{TP}^{\mathsf{min}}$ and $\mathsf{TP}^{\mathsf{avg}}$ over $\mathsf{PrXML}^{mux,det}_{cont}$ p-documents that are labeled with distributions in PP(*K*) is $\mathsf{FP}^{\#\mathsf{P}}$, since it is inherited from the $\mathsf{PrXML}^{cie}_{cont}$ case. Moreover, probability computation is of polynomial combined complexity for the class $\mathsf{SP}^{\mathsf{min}}$ and p-documents labeled with distributions from PP(*K*), as follows from Theorem 7.5.

Moments Computation. For TP^{α} where $\alpha \in \{\mathsf{min}, \mathsf{sum}, \mathsf{avg}\}$ over m-scalable p-documents the computations is in $\mathsf{FP}^{\#\mathsf{P}}$ as follows from Theorem 6.10. For $\mathsf{SP}^{\mathsf{min}}$ and $\mathsf{SP}^{\mathsf{sum}}$ over p-documents labeled with distributions in PP(*K*) we have polynomial-time combined complexity of moment computation as follows from Theorem 7.5.

9. APPROXIMATE COMPUTATION

As discussed in Sections 6 and 8, for several aggregate functions on PrXML^{cie} and $\mathsf{PrXML}^{mux,det}$ p-documents, membership, probability computation, and moment computation are hard. Fortunately, there are general sampling techniques which give randomized approximation algorithms for tackling intractability of computing the aforesaid quantities. We now discuss these techniques for discrete p-documents.

For instance, suppose we wish to consider the aggregate function countd on a p-document $\widehat{\mathcal{P}}$. In particular, say we are interested in approximating the probability $\mathsf{Pr}(\mathsf{countd}(\mathcal{P}) \leqslant 100)$. This probability can be estimated by drawing independent random samples of the document, and using the ratio of samples for which countd is at most 100 as an estimator. Similarly, if we wish to approximate $\mathbb{E}(\mathsf{countd}(\mathcal{P}))$, we can draw independent random samples and return the average of countd on the drawn samples.

The first important question is: is it possible at all to have a reasonably small number of samples to get a good estimation? It would not be helpful if an enormous number of samples is necessary. The good news is that the answer to the preceding question is "yes." The second question is: how many samples do we need? The following classical result helps us to answer both questions.

PROPOSITION 9.1 ([HOEFFDING 1963]). *Assume $U_1, \ldots, U_T$ are T independent identically distributed random variables, each of which takes values in an interval of width R and has mean μ. Let $\bar{U} := \frac{1}{T} \sum_{i=1}^{T} U_i$ be the empirical average. Then, for each $\varepsilon > 0$, we have* $\mathsf{Pr}(|\bar{U} - \mu| \geqslant \varepsilon) \leqslant 2 \exp\left(-\frac{2\varepsilon^2 T}{R^2}\right)$.

Additive vs. Multiplicative Error. Our task essentially reduces to the estimation of some quantity *Q*. Our goal is to return a number $\widetilde{Q}$ that is "close" to *Q*. There are two notions of measuring the error of estimation: *multiplicative* and *additive*.

For $\varepsilon > 0$, the estimation has *multiplicative error* ε if $|Q - \widetilde{Q}| \leqslant \varepsilon|Q|$. Such a notion appears in the database literature [Grädel et al. 1998], and Fully Polynomial-Time Randomized Approximation Schemes (FPTRAS) give guarantees based on this notion. Observe that in order for an estimate $\widetilde{Q}$ to achieve multiplicative error $\varepsilon \in]0, 1[$, the quantity Q is nonzero if and only if the estimate $\widetilde{Q}$ is nonzero. From the first statement of Theorem 6.2, we see that for p-documents $\widehat{\mathcal{P}} \in \mathsf{PrXML}^{cie}$ it is coNP-hard to decide whether $\Pr(\alpha(\mathcal{P}) = c)$ is zero, for any aggregate function α that distinguishes bags of elements and some value c. Hence, it follows that it is coNP-hard to give an estimation of $\Pr(\alpha(\mathcal{P}) = c)$ with multiplicative error ε, for $0 < \varepsilon < 1$.

In our applications, we consider the more tractable notion of *additive error* instead. For $\varepsilon > 0$, the estimation $\widetilde{Q}$ has *additive error* ε if $|Q - \widetilde{Q}| \leqslant \varepsilon$.

Sampling under PrXML^{cie} *and* $\mathsf{PrXML}^{mux,det}$. Under both models, sampling of a random tree can be performed easily. Under the PrXML^{cie} model, variables are sampled according to their probabilities and the corresponding tree is constructed. Under the $\mathsf{PrXML}^{mux,det}$ model, sampling is done by keeping all *det* nodes and a child for each *mux* node is chosen with the appropriate probability.

Approximating Membership. Suppose α is an aggregate function on some p-document $\widehat{\mathcal{P}}$, and we wish to know if the probability $\Pr(\alpha(\mathcal{P}) = x)$ is nonzero for some value x. The sampling algorithm is simple: we draw a certain number of samples and if for at least one sample, the event $\alpha(\mathcal{P}) = x$ happens, the algorithm returns *positive*, and *negative* otherwise. Proposition 9.1 gives a bound on the number of samples for which we can make a meaningful estimation.

COROLLARY 9.2. *For any aggregate function α, p-document $\widehat{\mathcal{P}} \in \mathsf{PrXML}^{cie,mux,det}$, any rational number x, and for any ε, $\delta > 0$, it is sufficient to have $O(\frac{1}{\varepsilon^2} \log \frac{1}{\delta})$ samples so that the following happens.*

(1) *If $\Pr(\alpha(\mathcal{P}) = x) = 0$, then* negative *is always returned.*
(2) *If $\Pr(\alpha(\mathcal{P}) = x) > \varepsilon$, then with probability at least $1 - \delta$,* positive *is returned.*

PROOF. The first part of the statement is trivial. For the second part, we let U_i be the Bernoulli variable that takes value 1 when $\alpha(\mathcal{P}) = x$ and 0 otherwise. It follows that $\mathbb{E}(U_i) = \Pr(\alpha(\mathcal{P}) = x)$. From Proposition 9.1, it is enough to have $O(\frac{1}{\varepsilon^2} \log \frac{1}{\delta})$ samples to estimate $\Pr(\alpha(\mathcal{P}) = x)$ with additive error at most ε and success probability at least $1 - \delta$. Since $\Pr(\alpha(\mathcal{P}) = x) > \varepsilon$, this means with desirable probability we would see at least one positive sample. □

For example, suppose we would like to see if the aggregate function min on some p-document $\widehat{\mathcal{P}} \in \mathsf{PrXML}^{cie}$ takes some specified value x with probability higher than ε. Then, for some fixed success probability, the number of samples depends only on ε, and is independent of the instance size. Note that if the probability is nonzero and below ε, the algorithm could return *negative*.

Approximating Distribution Points. Suppose α is an aggregate function on some p-document $\widehat{\mathcal{P}}$, I is a subset of the real numbers and we wish to approximate the probability $\Pr(\alpha(\mathcal{P}) \in I)$. For example, if $I = \{x\}$ for some number x, we may want to estimate the probability that the aggregate function evaluates to x; if $I =] - \infty, x]$, we estimate the probability that the aggregate function takes values at most x.

We sample instances of the p-document, and for each sample, let X_i be the corresponding value of the aggregate function. We let U_i to be the Bernoulli variable that takes value 1 when $X_i \in I$, and 0 otherwise. Then, it follows that $\mathbb{E}(U_i) = \Pr(\alpha(\mathcal{P}) \in I)$, and $\bar{U} := \frac{1}{T} \sum_{i=1}^{T} U_i$ is an estimate of $\Pr(\alpha(\mathcal{P}) \in I)$. Hence, we immediately obtain

the following result for approximating a point for the cumulative distribution of an aggregate function.

COROLLARY 9.3. *For any aggregate function α, p-document $\widehat{\mathcal{P}} \in \mathsf{PrXML}^{cie,mux,det}$, any subset I of the reals, and for any ε, $\delta > 0$, it is sufficient to have $O(\frac{1}{\varepsilon^2} \log \frac{1}{\delta})$ samples so that with probability at least $1 - \delta$, the quantity $\mathsf{Pr}(\alpha(\mathcal{P}) \in I)$ can be estimated with an additive error of ε.*

Observe that the number of samples in Corollary 9.3 is independent of the instance size. However, an additive error of ε would render the estimate useless if the probability to be estimated is less than ε. Hence, if we only care about probabilities above some threshold p_0, then it is enough to have the number of samples proportional to $1/p_0^2$ (with additive error, say, $p_0/10$).

Approximating Moments. Suppose f is some function on an aggregate function α, and we are interested in computing $\mathbb{E}(f(\alpha(\mathcal{P})))$. For each sample, we let $U_i := f(X_i)$, and compute the estimator $\bar{U} := \frac{1}{T} \sum_{i=1}^{T} U_i$.

COROLLARY 9.4. *Let $\widehat{\mathcal{P}} \in \mathsf{PrXML}^{cie,mux,det}$ be a p-document and f be a function on the aggregate function α such that $f(\alpha(\mathcal{P}))$ takes values in an interval of width R. Then, for any ε, $\delta > 0$, it is sufficient to have $O(\frac{R^2}{\varepsilon^2} \log \frac{1}{\delta})$ samples so that, with probability at least $1 - \delta$, the quantity $\mathbb{E}(f(\alpha(\mathcal{P})))$ can be estimated with an additive error of ε. In particular, if α takes values in $[0, R]$ and $f(\alpha) := \alpha^k$, then the k-th moment of $\alpha(\mathcal{P})$ around zero can be estimated with $O(\frac{R^{2k}}{\varepsilon^2} \log \frac{1}{\delta})$ samples.*

If the range R has magnitude polynomial in the problem size, then we have a polynomial-time algorithm. In our example for approximating $\mathbb{E}(\mathsf{countd}(\widehat{\mathcal{P}}))$, the range R can be at most the size of the problem instance. Hence, to estimate the expectation, it is enough to draw a quadratic number of random samples, and we have a polynomial-time approximation algorithm.

10. RELATED WORK

The literature about probabilistic relational databases is quite extensive. One of the early and seminal works is Barbará et al. [1992] where probabilities are associated with attribute values. Among the number of models and systems that have been proposed for representing and querying probabilistic data, one can distinguish between systems with limited expressive power such as the block-independent model [Dalvi and Suciu 2007] (which can be seen as a relational counterpart to $\mathsf{PrXML}^{mux,det}$) and the more complex, lineage-oriented, probabilistic database management systems like Trio [Widom 2005] and MayBMS [Koch 2009], that are closer in spirit to PrXML^{cie}. The latter are inspired by Imieliński and Lipski's c-tables [Imieliński and Lipski 1984] (though these are models for incomplete information, they can be applied to probabilistic information in a straightforward way[Green and Tannen 2006]).

The probabilistic XML models that have been proposed in the literature can be grouped in two main categories, depending on the kind of supported probabilistic dependencies: $\mathsf{PrXML}^{mux,det}$-like *local* dependencies [Nierman and Jagadish 2002; Hung et al. 2003; van Keulen et al. 2005; Hung et al. 2007], or PrXML^{cie}-like *global* dependencies [Abiteboul and Senellart 2006; Senellart and Abiteboul 2007]. A unifying framework for all these models, generalizing the distributional nodes introduced by Nierman and Jagadish [2002], has been proposed in Kimelfeld et al. [2008] and Abiteboul et al. [2009], which is the framework we use in this article. More recently Benedikt et al. [2010], a more expressive probabilistic XML model based on recursive

probabilistic processes, has been introduced to allow representing possible worlds of unbounded depth or width. That model is a generalization of PrXML^{mux,det}-like models, and an interesting extension of this work would be to understand which of the results presented in this article can be extended to that framework.

The complexity of nonaggregate query answering over PrXML^{mux,det} and PrXML^{cie} has been investigated in Kimelfeld and Sagiv [2007], Senellart and Abiteboul [2007], and Kimelfeld et al. [2008, 2009]. Several results presented here either extend or use these works. The dynamic-programming algorithm for computing the probability of a Boolean tree-pattern query from Kimelfeld and Sagiv [2007] and Kimelfeld et al. [2008, 2009] is in particular used for Claim 2 of Theorem 8.8. We study in more detail than in this work the case of nonaggregate tree-pattern queries with joins in Kharlamov et al. [2011]. The problem of tree-pattern query answering over PrXML^{mux,det} documents with constraints expressed using aggregate functions, that is, similar to the HAVING queries of SQL, is studied by Cohen et al. [2008]. We use the results of that article to prove Claim 2 of Theorem 8.7.

Only a few works have considered aggregate queries in a setting of incomplete data. In nonprobabilistic settings aggregate queries were studied for conditional tables [Lechtenbörger et al. 2002], for data exchange [Afrati and Kolaitis 2008], and for ontologies [Calvanese et al. 2008]. In probabilistic settings, to the best of our knowledge, in addition to the aforementioned [Cohen et al. 2008], only Ré and Suciu [2007] study aggregate queries. Ré and Suciu consider the problem of evaluating HAVING queries (using aggregate functions) in "block-independent databases," which are roughly PrXML^{mux,det} restricted to relations (limited-depth trees). The complexity bounds of Claim 3 of Theorem 8.7 use similar arguments to the corresponding results for block-independent databases presented in Ré and Suciu [2007]. In both Cohen et al. [2008] and Ré and Suciu [2007], the authors discuss the filtering of possible words that do not satisfy a condition expressed using aggregate functions, and do not consider the problem of computing the distribution of the aggregation, or moments thereof. Computation of the expected value of aggregate functions over a data stream of probabilistically independent data items is considered by Jayram et al. [2007]. This is a simpler setting than ours, but we use similar techniques in the proof of Theorem 8.8.

There is little earlier work on querying continuous probability distributions. Deshpande et al. [2004] build a (continuous) probabilistic model of a sensor network to run subsequent queries on the model instead of the original data. In Cheng et al. [2003], algorithms are proposed for answering simple classes of queries over uncertain information, typically given by a sensor network. As noted in a recent survey on probabilistic relational databases [Dalvi et al. 2009], "although probabilistic databases with continuous attributes are needed in some applications, no formal semantics in terms of possible worlds has been proposed so far." We proposed in this article such a formal semantics.

11. CONCLUSION

We provided algorithms for and a characterization of the complexity of computing aggregate queries for both PrXML^{mux,det} and PrXML^{cie} models, that is, very general probabilistic XML models. We also considered the expected value and other moments, that is, summaries of the probability distribution of the results of aggregate functions. In the case of PrXML^{mux,det}, we have identified a fundamental property of aggregate functions, that of being *monoid*, that entails tractability. The complexity of aggregate computations in many cases has led us to introduce polynomial-time randomized approximation schemes. Finally, a last original contribution has been the definition of a formal continuous extension of probabilistic XML models for which the results of the discrete case can be adapted.

In summary, and abstracting out the details, answering aggregate queries over probabilistic XML is tractable in data complexity if and only if the four following conditions are satisfied:

—the query language does not involve joins;
—the probabilistic data model does not involve global dependencies (*cie* nodes);
—the aggregate functions used are monoid functions with a domain whose size is bounded in the input size;
—continuous distributions present in the data can be tractably convoluted, summed, integrated, multiplied (e.g., this is the case for $\mathsf{PP}(K)$).

Our work can be extended in a number of directions. First, we intend to implement a system that manages imprecise data with aggregate functions. In particular, we want the system to handle continuous probabilities, which are quite useful in practice. Second, it should be possible to extend our results on aggregate queries to the wide and deep models of probabilistic XML from Benedikt et al. [2010], which is conceptually a generalization of the *mux-det* model. Third, note that the continuous distributions attached to leaves are essentially independent one from the other. This means, for instance, that $\mathsf{PrXML}_{cont}^{cie,mux,det}$ is not a *strong representation system* [Abiteboul et al. 1995] for the language of aggregate tree-pattern queries, or even just for the language of tree-pattern queries with inequalities: it is impossible to represent the output of a query (or the result of an update based on this query language) in the same framework. It would be interesting to study extensions of the model that are strong representation systems (for instance, by annotating the p-document leaves with algebraic expressions over random variables with values in a continuous space).

REFERENCES

ABITEBOUL, S., CHAN, T.-H. H., KHARLAMOV, E., NUTT, W., AND SENELLART, P. 2010. Aggregate queries for discrete and continuous probabilistic XML. In *Proceedings of the International Conference on Database Theory (ICDT)*. ACM, New York, 50–61.

ABITEBOUL, S., HULL, R., AND VIANU, V. 1995. *Foundations of Databases*. Addison-Wesley, Reading, PA.

ABITEBOUL, S., KIMELFELD, B., SAGIV, Y., AND SENELLART, P. 2009. On the expressiveness of probabilistic XML models. *VLDB J. 18,* 5, 1041–1064.

ABITEBOUL, S. AND SENELLART, P. 2006. Querying and updating probabilistic information in XML. In *Proceedings of the International Conference on Extending Database Technology (EDBT)*. ACM, New York.

AFRATI, F. N. AND KOLAITIS, P. G. 2008. Answering aggregate queries in data exchange. In *Proceedings of the ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS)*. ACM, New York.

ASH, R. B. AND DOLÉANS-DADE, C. A. 2000. *Probability & Measure Theory*. Academic Press, San Diego, CA.

BARBARÁ, D., GARCIA-MOLINA, H., AND PORTER, D. 1992. The management of probabilistic data. *Trans. Knowl. Data Engin. 4,* 5, 487–502.

BENEDIKT, M., KHARLAMOV, E., OLTEANU, D., AND SENELLART, P. 2010. Probabilistic XML via Markov chains. *Proc. VLDB Endowm. 3,* 1, 770–781.

CALVANESE, D., KHARLAMOV, E., NUTT, W., AND THORNE, C. 2008. Aggregate queries over ontologies. In *Proceedings of the International Workshop on Ontologies and Information Systems for the Semantic Web (ONISW)*. ACM, New York.

CHENG, R., KALASHNIKOV, D. V., AND PRABHAKAR, S. 2003. Evaluating probabilistic queries over imprecise data. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*. ACM, New York.

COHEN, S., KIMELFELD, B., AND SAGIV, Y. 2008. Incorporating constraints in probabilistic XML. In *Proceedings of the ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS)*. ACM, New York.

COHEN, S., KIMELFELD, B., AND SAGIV, Y. 2009. Running tree automata on probabilistic XML. In *Proceedings of the ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS)*. ACM, New York.

COHEN, S., SAGIV, Y., AND NUTT, W. 2006. Rewriting queries with arbitrary aggregation functions using views. *Trans. Datab. Syst. 31,* 2, 672–715.

DALVI, N., RÉ, C., AND SUCIU, D. 2009. Probabilistic databases: Diamonds in the dirt. *Comm. ACM 52,* 7, 86–94.

DALVI, N. N. AND SUCIU, D. 2007. Management of probabilistic data: Foundations and challenges. In *Proceedings of the ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS)*. ACM, New York.

DESHPANDE, A., GUESTRIN, C., MADDEN, S., HELLERSTEIN, J. M., AND HONG, W. 2004. Model-Driven data acquisition in sensor networks. In *Proceedings of the International Conference on Very Large Data Bases (VLDB)*. Morgan Kaufmann, San Fransisco, CA.

FRIEDLANDER, F. G. AND JOSHI, M. 1999. *Introduction to the Theory of Distributions* 2nd Ed. Cambridge University Press, Cambridge, UK.

GAREY, M. R. AND JOHNSON, D. S. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, New York.

GRÄDEL, E., GUREVICH, Y., AND HIRSCH, C. 1998. The complexity of query reliability. In *Proceedings of the ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS)*. ACM, New York, 227–234.

GREEN, T. J. AND TANNEN, V. 2006. Models for incomplete and probabilistic information. In *Proceedings of the International Conference on Extending Database Technology (EDBT) Workshops IIDB*. ACM, New York.

HOEFFDING, W. 1963. Probability inequalities for sums of bounded random variables. *J. Amer. Statist. Assoc. 58,* 301, 16–30.

HUNG, E., GETOOR, L., AND SUBRAHMANIAN, V. S. 2003. PXML: A probabilistic semistructured data model and algebra. In *Proceedings of the International Conference on Data Engineering*. IEEE, Los Alamitos, CA.

HUNG, E., GETOOR, L., AND SUBRAHMANIAN, V. S. 2007. Probabilistic interval XML. *Trans. Comput. Logic 8,* 4.

IMIELIŃSKI, T. AND LIPSKI, W. 1984. Incomplete information in relational databases. *J. ACM 31,* 4, 761–791.

JAYRAM, T. S., KALE, S., AND VEE, E. 2007. Efficient aggregation algorithms for probabilistic data. In *Proceedings of the Symposium on Discrete Algorithms (SODA)*. SIAM, Philadelphia, PA.

KHARLAMOV, E. 2011. A probabilistic approach to XML data management. Ph.D. thesis, KRDB Research Centre, Faculty of Computer Science, Free University of Bozen-Bolzano. http://www.inf.unibz.it/ kharlamov/.

KHARLAMOV, E., NUTT, W., AND SENELLART, P. 2011. Value joins are expensive over (probabilistic) XML. In *Proceedings of the International Conference on Extending Database Technology (EDBT) Workshops, LID*. ACM, New York.

KIMELFELD, B., KOSHAROVSKY, Y., AND SAGIV, Y. 2008. Query efficiency in probabilistic XML models. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*. ACM, New York.

KIMELFELD, B., KOSHAROVSKY, Y., AND SAGIV, Y. 2009. Query evaluation over probabilistic XML. *VLDB J. 18,* 5, 1117–1140.

KIMELFELD, B. AND SAGIV, Y. 2007. Matching twigs in probabilistic XML. In *Proceedings of the International Conference on Very Large Data Bases (VLDB)*. ACM, New York.

KOCH, C. 2009. MayBMS: A system for managing large uncertain and probabilistic databases. In *Managing and Mining Uncertain Data*, C. Aggarwal, Ed., Springer, New York.

LECHTENBÖRGER, J., SHU, H., AND VOSSEN, G. 2002. Aggregate queries over conditional tables. *J. Intell. Inf. Syst. 19,* 3, 343–362.

NIERMAN, A. AND JAGADISH, H. V. 2002. ProTDB: Probabilistic data in XML. In *Proceedings of the International Conference on Very Large Data Bases (VLDB)*. Morgan Kaufmann, San Fransisco, CA.

PAPADIMITRIOU, C. H. 1994. *Computational Complexity*. Addison Wesley, Reading, PA.

PROVAN, J. S. AND BALL, M. O. 1983. The complexity of counting cuts and of computing the probability that a graph is connected. *SIAM J. Comput. 12,* 4, 777–788.

RÉ, C. AND SUCIU, D. 2007. Efficient evaluation of HAVING queries on a probabilistic database. In *Proceedings of the Database Programming Languages*. Springer, New York.

SENELLART, P. AND ABITEBOUL, S. 2007. On the complexity of managing probabilistic XML data. In *Proceedings of the ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS)*. ACM, New York.

VAN KEULEN, M., DE KEIJZER, A., AND ALINK, W. 2005. A probabilistic XML approach to data integration. In *Proceedings of the International Conference on Data Engineering*. IEEE, Washington, DC.

WIDOM, J. 2005. Trio: A system for integrated management of data, accuracy, and lineage. In *Proceedings of the Conference on Innovative Data Systems Research (CIDR)*. Online Proceedings.