Modeling, Querying, and Mining Uncertain XML Data

Evgeny Kharlamov
Free University of Bozen-Bolzano, Italy;
INRIA Saclay, France

Pierre Senellart
Télécom ParisTech, France

This chapter deals with data mining in uncertain XML data models, this uncertainty typically coming from imprecise automatic processes. We first review the literature on modeling uncertain data, starting with well-studied relational models and moving then to their semistructured counterparts. We focus on a specific probabilistic XML model, that allows representing arbitrary finite distributions of XML documents, and has been extended to also allow continuous distributions of data values. We summarize previous work on querying this uncertain data model and show how to apply the corresponding techniques to several data mining tasks, exemplified through use cases on two running examples.

1 Introduction

Though traditional database applications, for instance, bank account management, have no room for uncertainty, more recent applications, such as information extraction from the Web, automatic schema matching in information integration, or information gathering from sensor networks are inherently imprecise. This uncertainty is sometimes represented as the *probability* that the data is correct, as with conditional random fields in information extraction (Lafferty, McCallum, & Pereira, 2001), or uncertain schema mappings in information integration (Dong, Halevy, & Yu, 2009). In other cases, only *confidence* in the information is provided by the system, which can be seen after renormalization as an approximation of the probability. More rarely, some applications do not provide any form of preference among possible uncertain choices (think, for example, of missing data in a data recovery application), or only some unweighted preferences (like the core solution in data exchange (Fagin, Kolaitis, & Popa, 2005) or a minimal repair in managing inconsistent databases (Chomicki & Libkin, 2000; Lopatenko & Bertossi, 2007)).

Usually, data uncertainty is not formally taken into account: only the most likely interpretation is kept for future processing, or all probable choices above a threshold are maintained. We claim this is not sufficient. There is a need for managing the imprecision in this data more rigorously. The need is even stronger when the uncertain data is manipulated by other systems, potentially uncertain themselves. A good example of that is data mining. Consider a scenario

where some dataset (say, a list of emails) was acquired, cleaned, and enriched, by a variety of systems (information extraction, deduplication, data integration, natural language analysis, sentiment analysis, etc.). We now want to mine this dataset, for instance to construct from it a list of popular keywords, or to build a social network of individuals, where the friendship links between two persons is derived from their recorded interactions. An application that would make use of the inherent uncertainty in the dataset would be able to discover much more knowledge than one that would ignore it altogether. Besides, in the mining task the confidence annotation in the data could also be used to derive the confidence of the resulting (mined) data.

A number of models and systems for managing uncertain data have been proposed in the literature and a high-level picture of some of them is presented in this chapter. We focus, however, on the particular case of XML data, adapted in the cases where the information is either not strictly constrained by a schema (e.g., Web data), or inherently tree-like (mailing lists, parse trees of natural language sentences, etc.). We also mostly discuss probabilistic models, which have the advantage, in addition to being suited to a number of tasks that provide probability or probability-like confidence scores, of allowing extensive mathematical manipulations (more so than models based on fuzzy logic (Galindo, Urrutia, & Piattini, 2005), that are not discussed in this chapter).

The objective of our chapter is thus to bridge the studies on uncertain XML and data mining. On the one hand, we want to introduce different models of uncertain data to the data mining community. On the other hand, we want to study different data mining tasks for probabilistic XML. Recent studies of probabilistic XML (Abiteboul, Kimelfeld, Sagiv, & Senellart, 2009; Kimelfeld, Kosharovsky, & Sagiv, 2009; Kharlamov, Nutt, & Senellart, 2010) focus on query answering and updates, but mining, that has been studied in the context of relational probabilistic data (Aggarwal, 2009; Bernecker, Kriegel, Renz, Verhein, & Züfle, 2009), has not received attention in the semistructured case. Note that the change of representation format from tables to trees also makes data mining tasks different (Nayak, 2005). In this chapter we propose methods for mining probabilistic XML data (frequent items, correlations, summaries of data values, etc.) that rely on the existing literature on probabilistic XML querying (Kimelfeld et al., 2009; Abiteboul, Chan, Kharlamov, Nutt, & Senellart, 2010).

In the following part of this chapter we discuss several main approaches to uncertainty modeling. We start with uncertain relational databases and present examples and intuitions of incomplete and probabilistic tables. We discuss how these approaches were adapted to the semistructured setting and illustrate incomplete XML trees and two probabilistic XML models: with local and global probabilistic relationships. The next section is devoted to a formal presentation of these probabilistic XML models; we present the syntax and semantics of discrete and continuous probabilistic XML. We then summarize known results about probabilistic XML querying, both for Boolean and aggregate queries, that are at the basis of the data mining approaches we present in a subsequent section, where we give examples and develop computation techniques for mining frequent, co-occurring, or popular items, or for summarizing continuous distributions.

2 Models of Uncertainty

In this section, we discuss how to formally model the uncertainty that arises in applications. Unsurprisingly, there is a trade-off between expressiveness and succinctness of the model on one hand and its simplicity and the ability to efficiently use it for query processing on the other hand. We first discuss relational data then see how these relational uncertain data models can be adapted to XML, which we will focus on in the rest of this chapter. Historically, uncertain relational data models precede semistructured ones and these heavily rely on the ideas developed for the relational setting. Moreover, translating probabilistic XML into relations is a way to manage it, as shall be discussed further. Due to this historical and practical importance of relational uncertain models, we present them in detail.

2.1 Relational Models of Uncertain Data

The relational model, proposed by Codd (1970) is the most commonly used data representation model today. The need for modeling uncertainty in relational tables has been felt as early as the mid-1970s (Codd, 1975) and has led to the notion of *null* values implemented in System R, one of the first relational DBMS, a few years afterwards (Date, 1981). We first present this way of representing uncertain data and look next to more expressive models.

Codd tables. The first works mentioning the problem of incompleteness in relational models are by Codd (1975, 1979). Codd proposed to augment relational tables with special constants, called *nulls* and usually denoted @, assumed to be different from all other data values. A null is meant to be a syntactic substitute for a missing or uncertain value. Relational tables with tuples that may contain @ are called *Codd tables* or *naïve tables* (Abiteboul, Hull, & Vianu, 1995). For example, one can use @ as a value for the unknown telephone number of Mary in the following Codd table T_{ex}:

id	name	tel_nr
1	Mary	@

Codd proposed a semantics for query evaluation over tables with nulls based on three-valued logics. According to this semantics, logical expressions involving nulls are evaluated using "true", "false" and "unknown", e.g., $(@ > 5)$ is unknown, $(@ > 5) \wedge (5 < 3)$ is false while $(@ > 5) \vee (3 < 5)$ is true. Codd's semantics for nulls is the current standard in SQL (ISO/IEC, 2008) and implemented in the main relational DBMSs. However, it can produce counterintuitive results, as noted by Grant (1977): evaluating a query

```
SELECT * FROM T_ex WHERE tel_nr = '333 111' OR tel_nr <> '333 111'
```

over T_{ex} returns no answers, while one might expect the tuple about Mary.

An alternative semantics of Codd tables, that will be helpful for other uncertain data models as well, is given by *sets of possible worlds* (Abiteboul et al., 1995). From the incomplete data about Mary in our example we can only guess what her missing phone number is. Each guess gives one completion of the data, that is, one complete database, and all the possible guesses give a set of data completions and, consequently, a set of corresponding databases, or *worlds*. When

the *domain* (the set of legal values) for a given attribute is infinite, there can be infinitely many such worlds. The actual database is assumed to be one of these possible worlds. In the following we use the term *incomplete database* to refer to such a set of possible worlds, denoted as $\mathscr{D}$. Formalisms to represent sets of possible databases, such as Codd tables, are called *representation systems*.

From the point of view of possible-world semantics, nulls are treated as variables, and different occurrences of a null value correspond to different variables. The semantics of a Codd table T is the set of all databases $rep(T)$ (where *rep* stands for *represents*) obtained from it by substituting every occurrence of @ with a data value from the corresponding domain. Looking back to the example of Mary,

$$rep(T_{\text{ex}}) = \{\{1, \text{Mary}, 111\ 333\}, \{1, \text{Mary}, 333\ 444\} \ldots\}.$$

Codd tables have serious limitations in representing incomplete databases, in particular when one wants to model additional knowledge about the unknown data values (Imieliński & Lipski, 1984). Consider the following incomplete database: Mary and John, spouses, have the same unknown telephone number, which might be different from the unknown telephone number of a third person Bob. In addition we may have constraints on the unknown values, for instance, that Mary's age is between 30 and 35 and that she is younger than John and older than Bob. This incomplete database cannot be represented as a Codd table, since we need the ability to have coreferences across null values, and to express constraints on unknown values. To overcome these limitations, Imieliński & Lipski (1984) introduced the representation system of *c-tables* ("c" stands for "conditional").

C-tables. In c-tables, nulls are labeled with subscripts such that two nulls with the same label always denote the same value. C-tables are also equipped with an extra column to store (local) boolean conditions of the form $@_1 < c$ or $@_1 < @_2$ on labeled nulls and constants. Here is an example of a c-table representing the incomplete database discussed in the previous paragraph:

id	name	tel_nr	age	cond
1	Mary	$@_1$	$@_3$	$@_3 \in (30,35)$
2	John	$@_1$	$@_4$	$@_3 < @_4$
3	Bob	$@_2$	$@_5$	$@_5 < @_3$

Note that a c-table where every labeled null occurs only once and with an empty condition column is a Codd table. The possible-world semantics of c-tables is similar to that of Codd tables, with the difference that one substitutes all occurrences of the same labeled null with the same constant, and afterwards deletes from the resulting table each tuple whose attributes do not satisfy the conditions associated to the tuple.

Evaluation of a query Q over an incomplete database $\mathscr{D}$ is defined point-wise: one queries all the databases in $\mathscr{D}$ separately and then considers the set of query results, which is an incomplete database, as the output $Q(\mathscr{D})$, that is, $Q(\mathscr{D}) = \{Q(d) \mid d \in \mathscr{D}\}$. An important question studied for different representation systems is to understand under which query languages they are *closed*, i.e., given a representation T and a query Q, whether it is possible to represent $Q(rep(T))$ in the same formalism as T (Sarma, Benjelloun, Halevy, & Widom, 2006). It was shown (Imieliński

& Lipski, 1984) that Codd tables are closed under the *project-union* fragment of the relational algebra, but are not closed under *select* or *join*. In contrast, c-tables are closed under the whole relational algebra, i.e., selection, projection, join, difference, union, and (attribute) renaming. Interestingly, these closure results also give polynomial-time algorithms for directly constructing a representation of $Q(rep(T))$ by evaluating Q over T, treating nulls as constants, and (for c-tables) combining the conditions with Boolean operators that depend on the operation performed over the corresponding tuples. Details can be found in (Imieliński & Lipski, 1984).

Probabilistic models. Modeling uncertainty by incomplete databases is not enough: it is often useful to have some indication of the likelihood of possible worlds, or of the confidence we have in a piece of information. This can be done by adding probabilities to incomplete database models. A *probabilistic database* is a probabilistic distribution over a set of possible worlds. As for incomplete databases, we are interested in compact representation systems for such probabilistic distributions. We are thus looking for probabilistic counterparts to the incomplete database representation systems described earlier.

A first model, inspired by Codd tables, is that of *p-Codd tables* (Lakshmanan, Leone, Ross, & Subrahmanian, 1997). They assume that each attribute inside each tuple has a separate finite distribution of values. Consider a tuple whose value of the `name` attribute is either "Mary" with probability $p = 0.3$ or "John" with $p = 0.7$; the phone number is either "111 333" with $p = 0.2$, or "333 444" with $p = 0.8$. The corresponding p-Codd table can be represented as follows:

`id`	`name`	`tel_nr`
1	Mary ($p = 0.3$)	111 333 ($p = 0.2$)
	John ($p = 0.7$)	333 444 ($p = 0.8$)

Every world of a p-Codd table has an associated probability that is equal to the product of the probabilities of every choice taken to obtain that world. For example, the probability of the world $\{(1, \text{Mary}, 111\ 333)\}$ is $0.3 \times 0.2 = 0.06$.

Another early probabilistic model has been independently introduced by Fuhr & Rölleke (1997) and Zimányi (1997). A so-called *tuple-independent* database is a probabilistic database where the probability of each tuple is given, and where the existence of a tuple is independent of the existence of the other tuples. The model was further extended by Ré, Dalvi, & Suciu (2006) to the so-called *block-independent* databases or *BID*, by regrouping tuples of a relation into independent blocks, with mutually exclusive tuples inside a given block, as in the following example:

`id`	`name`	`age`	`prob.`
1	Mary	30	0.3
		32	0.5
		35	0.2
2	John	37	0.5
		40	0.5

The semantics of BIDs chooses at most one tuple in each block, and choices across blocks are independent. In this example, the tuple $(1, \text{Mary}, 30)$ can be kept with probability 0.3, the tuple $(1, \text{Mary}, 32)$ with 0.5, and $(1, \text{Mary}, 35)$ with 0.2, and these tuples are mutually exclusive.

Independently from this choice, one of the two tuples $(2, \text{John}, 37)$ or $(2, \text{John}, 40)$ can be chosen with uniform probability.

Both p-Codd tables and BIDs assume that occurrences of attribute values or tuples in a given world are independent from each other. Due to this assumption the models cannot capture complex probabilistic dependencies across data items. This limitation is similar to the limitations of Codd tables with respect to c-tables. A very general probabilistic model is thus an extension of c-tables to *probabilistic c-tables* (Green & Tannen, 2006), where every variable in a table is treated as a random variable.

Recently a number of systems to support probabilistic relational databases were developed, they include *Trio* at Stanford (Widom, 2005), *MYSTIQ* at U. Washington (Boulos et al., 2005), *MayBMS* at Cornell and U. Oxford (Huang, Antova, Koch, & Olteanu, 2009), and *PrDB* at U. Maryland (Sen, Deshpande, & Getoor, 2009). The Trio system implements probabilistic variants of *or-sets* (Imieliński, Naqvi, & Vadaparty, 1991; Libkin & Wong, 1996), and *?-sets* (Sarma et al., 2006), which can be seen as c-tables with correlations within tuples only. In addition, Trio supports data provenance in order to trace the origin of query results. MYSTIQ implements BIDs. MayBMS is an extension of PostgreSQL and implements so called *world-set decompositions* (Antova, Koch, & Olteanu, 2007), that are probabilistic c-tables where the domains of labeled nulls are finite and dependencies (conditions) on variables and data items are of a limited form. The dependencies are encoded using the cross product of probabilistic tables. PrDB is based on graphical models to capture uncertain data, which are a well known probabilistic modeling technique coming from the statistics and machine learning community. PrDB captures both tuple and attribute uncertainty together with complex probabilistic correlations among tuples and attributes, encoded as factored join distributions.

For further reading on probabilistic databases we refer the reader to (Green & Tannen, 2006; Sarma et al., 2006), where there is a nice overview of some probabilistic models.

2.2 Semistructured Models of Uncertain Data

Semistructured models of uncertain data have been much less studied than relational ones. As we shall see, the models proposed so far are, unsurprisingly, semistructured counterparts of the relational models discussed above. However, they are still worthwhile to study on their own, since they are used to represent different kinds of uncertain data and that typical query languages over trees have different expressive power from that over relations (tree-pattern queries vs conjunctive queries, XPath vs SQL).

Incomplete XML. Barceló, Libkin, Poggi, & Sirangelo (2009) proposed a model of incomplete XML that is inspired by c-tables and tree-pattern queries. In the spirit of c-tables, their model allows representing unknown labels, using variables. Besides that, their model also supports structural uncertainty, which is not possible in c-tables, but is common when data is queried through tree-patterns queries. It is thus possible to model uncertainty along horizontal (sibling) and vertical (descendant, child) navigational axes. To illustrate, consider the uncertain XML document about people in a company of Figure 1. The first person is a manager with name Mary and unknown phone number, denoted x. The second person is John, his position is unknown, denoted as the variable Y, and his telephone is unknown but the same as of Mary. The third person

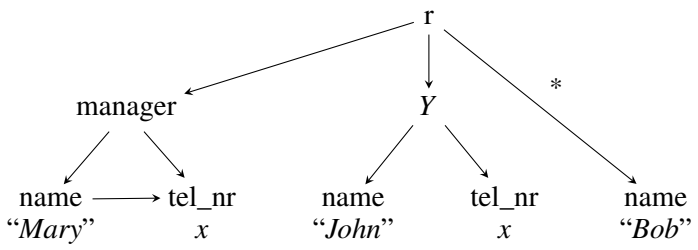

Figure 1: An incomplete XML document

has name Bob. It is also known that (the node storing) Mary's phone number is the following sibling of (the node storing) her name and (the node storing) Bob's name is a descendant of the root. No further structural information is known.

Problems over uncertain XML documents studied in (Barceló et al., 2009) are consistency of partial descriptions (i.e., whether there is an XML document satisfying a given description), representability of complete documents by incomplete ones, and query answering. They also show how schema information, the presence of node identifiers, and missing structural information affect the complexity of these main computational problems, and they identify tractable query classes over incomplete XML descriptions.

Probabilistic XML. Similarly as in the relational setting, a probabilistic semistructured database is a probability distribution over regular XML documents. A number of models (Nierman & Jagadish, 2002; Hung, Getoor, & Subrahmanian, 2003a, 2003b; Keulen, Keijzer, & Alink, 2005; Abiteboul & Senellart, 2006) have been proposed for compact representation of probabilistic XML databases. The model we focus here is called PrXML. It was introduced in (Kimelfeld, Kosharovski, & Sagiv, 2008; Abiteboul et al., 2009) and it uses *p-documents* as representations. Practically all probabilistic XML models proposed in the literature can be defined by means of p-documents.

P-documents are trees with two types of nodes: *ordinary* and *distributional*. A p-document can be thought of as a probabilistic process that generates a random XML document in a conceptually simple way, namely, each distributional node chooses a subset of its children (see next section). Therefore, each distributional node of a p-document should specify the probability distribution of choosing a subset of its children in the above random process. There are several types of distributional nodes that differ from one another in how they specify probabilities. An example of a p-document is presented in Figure 2. This p-document, denoted $\widehat{\mathscr{P}}_{\text{MBOX-L}}$, represents a mailbox organized in threads. It contains one (uncertain) thread with two (uncertain) messages. The possible worlds of this probabilistic XML document follow the DTD of Figure 3 (left). The following fragments of data are uncertain in $\widehat{\mathscr{P}}_{\text{MBOX-L}}$: the recipient of the first message, the sender of the second, and a part of the content of the second message. We will come back to this example in the next section and explain it in more detail.

We consider four types of distributional nodes:

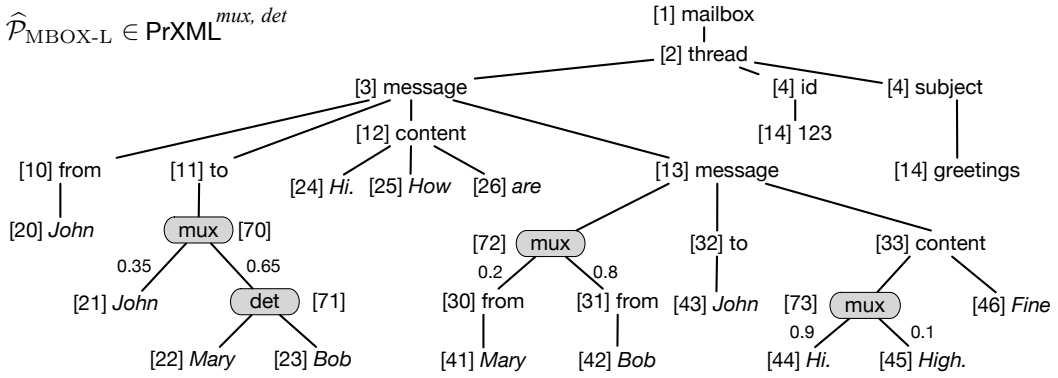

Figure 2: $\mathsf{PrXML}^{\mathsf{mux,det}}$ p-document $\widehat{\mathscr{P}}_{\text{MBOX-L}}$: Mailbox organized in threads

det for deterministic (every child is chosen with probability 1);

ind for independent (every child can be chosen independently from another, according to the corresponding assigned probability);

mux for mutually exclusive (at most one child can be chosen);

cie for conjunction of independent events (each child is chosen according to a conjunction of probabilistically independent events, which can be used globally throughout the p-document).

It may seem that using det nodes is redundant, but actually they increase expressive power when used in conjunction with some of the other types.

We define different families of p-documents in terms of the types of distributional nodes that are allowed. PrXML^{C}, where $C \subseteq \{\mathsf{mux}, \mathsf{ind}, \mathsf{det}, \mathsf{cie}\}$, denotes the family of p-documents that use the types appearing in the subset C. We ignore here the exp distributional nodes from (Abiteboul et al., 2009) that are a generalization of mux nodes with less practical interest.

$\mathsf{PrXML}^{\mathsf{mux,ind}}$ is a local probabilistic model, that is, it allows for dependencies between siblings only, or, in other words, for hierarchical dependencies. In this respect $\mathsf{PrXML}^{\mathsf{mux,ind}}$ is a semistructured counterpart of BIDs. It is easy to see that any $\mathsf{PrXML}^{\mathsf{mux,ind}}$ p-document can actually be transformed into an equivalent and as succinct $\mathsf{PrXML}^{\mathsf{mux,det}}$ p-document. Therefore, we shall ignore in the following ind distributional nodes. On the other hand, $\mathsf{PrXML}^{\mathsf{cie}}$ is global since it allows for complicated probabilistic relationships between data items. Thus, $\mathsf{PrXML}^{\mathsf{cie}}$ is a semistructured counterpart of probabilistic c-tables.

The PrXML data model has been extended to represent continuous distributions of data values (Abiteboul et al., 2010), and to capture infinite probability spaces of documents where the depth or the width of the possible worlds is unbounded (Benedikt, Kharlamov, Olteanu, & Senellart, 2010). We focus in this chapter on the standard PrXML model together with continuous distributions, that we formally introduce in the next section.

An alternative probabilistic XML representation system is that of Cohen, Kimelfeld, & Sagiv (2008). The idea is to add, on top of a simple PrXML description, external *constraints* (defined by

Mailbox DTD	
mailbox:	$(\text{thread})^*$
thread:	(message, id, subject)
message:	$(\text{from}, \text{to}, \text{content}, \text{message}^*)$
from:	#PCDATA
to:	#PCDATA
content:	#PCDATA
subject:	#PCDATA

Electricity Consumption DTD	
electr-cons:	(room1, room2)
room1:	$(\text{measurement})^*$
room2:	$(\text{measurement})^*$
measurement:	(date, value)
date:	#PCDATA
value:	#PCDATA

Figure 3: On the left: Mailbox DTD; on the right: Electricity Consumption DTD

aggregate tree-pattern queries) restricting the validity of possible worlds. For instance, constraints may only allow possible worlds where at least n nodes of a given type are present. This is related to the coupling of probabilistic XML documents with schema validation, discussed in (Cohen, Kimelfeld, & Sagiv, 2009).

3 Probabilistic XML

In this section, we present more formally the syntax and semantics of the PrXML model. We first focus on a discrete model (to represent discrete probability distributions) and then extend it to allow continuous data values.

3.1 Discrete Probabilistic XML

We model XML documents as unranked, unordered, labeled trees. Not taking into account the order between sibling nodes in an XML document is a common but non-crucial assumption. The same modeling can be done for ordered trees, without much change to the theory. A *finite probability space* over documents, px-*space* for short, is a pair $(\mathscr{D}, \mathsf{Pr})$, where $\mathscr{D}$ is a finite set of documents and Pr maps each document to a probability $\mathsf{Pr}(d)$ such that

$$\sum\{\mathsf{Pr}(d) \mid d \in \mathscr{D}\} = 1.$$

p-Documents: Syntax. The PrXML model from (Kimelfeld et al., 2009; Abiteboul et al., 2009) uses *p-documents* to represent px-spaces in a compact way. As already discussed, a p-document is similar to a document, with the difference that it has two types of nodes: ordinary and distributional. Distributional nodes are used for defining the probabilistic process that generates random documents but they do not actually occur in these. 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 $\mathscr{X}$ of independent Boolean random variables with some specified probability distribution Δ over them. A *p-document*, denoted by $\widehat{\mathscr{P}}$, is an unranked, unordered, labeled tree. Each node has a unique identifier u and a label $\mu(u)$ in $\mathscr{L} \cup \{\mathsf{cie}(E)\}_E \cup \{\mathsf{mux}(\mathsf{Pr})\}_{\mathsf{Pr}} \cup \{\mathsf{det}\}$ where $\mathscr{L}$ are labels of *ordinary* nodes, and the others are

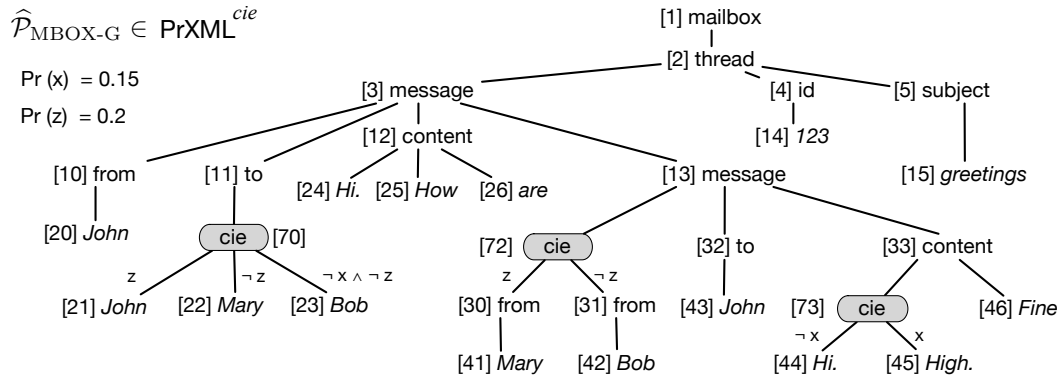

Figure 4: PrXML$^{\text{cie}}$ p-document $\widehat{\mathscr{P}}_{\text{MBOX-G}}$: Mailbox organized in threads

labels of *distributional* nodes. We consider three kinds of the latter labels: $\mathsf{cie}(E)$ (for conjunction of independent events), $\mathsf{mux}(\mathsf{Pr})$ (for mutually exclusive), and det (for deterministic). If a node u is labeled with $\mathsf{cie}(E)$, then E is a function that assigns to each child of u a conjunction $e_1 \wedge \cdots \wedge e_k$ of literals (x or $\neg x$, for $x \in \mathscr{X}$). If u is labeled with $\mathsf{mux}(\mathsf{Pr})$, then Pr assigns to each child of u a probability with the sum across children at most 1.

Example 1 *Two p-documents are shown in Figures 2 and 4.*

The first one belongs to PrXML$^{\text{mux,det}}$ *since it has only* mux *and* det *distributional nodes. The p-document stores one thread with two messages and the subject "greetings". The first message is sent by John and addressed to either himself or to both Mary and Bob. The reason why we may have this uncertainty in the recipient is that we try to identify the persons' name using email addresses. We know that John sent a message to two email addresses that either both belong to John himself with the probability* $\mathsf{Pr}(n_{21}) = 0.35$*, or to Mary and Bob with the probability* $\mathsf{Pr}(n_{71}) = 0.65$*. This probabilistic ambiguity is modeled with the* mux *distributional node* n_{70}*, since the two options are represented as sub-documents rooted at this node. Note that the* det *distributional node* n_{71} *is needed to account for both Mary and Bob as the second probabilistic option for the recipient. Another kind of uncertainty is in the content of the second message. This message contains a voice record for which automatic speech recognition is ambiguous. In* $\widehat{\mathscr{P}}_{MBOX\text{-}G}$ *under the node* n_{73} *we have two possible interpretations of a word from a voice record: "Hi" and "High". In our example the interpretation "Hi" has a higher probability, i.e.,* 0.85*.*

The second p-document, $\widehat{\mathscr{P}}_{MBOX\text{-}G}$*, belongs to* PrXML$^{\text{cie}}$ *since it has only* cie *distributional nodes. For example, node* n_{70} *has the label* $\mathsf{cie}(E)$ *and three children* n_{21}*,* n_{22} *and* n_{23}*, such that the event formulas labeling the nodes are, respectively,* $E(n_{21}) = z$*,* $E(n_{22}) = \neg z$ *and* $E(n_{23}) = \neg x \wedge \neg z$*. The probabilities of the event variables are* $\mathsf{Pr}(x) = 0.15$ *and* $\mathsf{Pr}(z) = 0.2$*.* ■

p-Documents: Semantics. The *semantics of a p-document* $\widehat{\mathscr{P}}$, denoted by $[\![\widehat{\mathscr{P}}]\!]$, is a px-space over documents, where the documents are obtained from $\widehat{\mathscr{P}}$ following a randomized three-step process (Abiteboul et al., 2009).

1. We choose a valuation ν of the variables in $\mathscr{X}$. The probability of this choice, according to the distribution Δ, is

$$p_\nu = \prod_{\substack{x \in X \\ \nu(x)=\mathsf{true}}} \Delta(x) \cdot \prod_{\substack{x \in X \\ \nu(x)=\mathsf{false}}} (1-\Delta(x)).$$

2. For each cie node labeled $\mathsf{cie}(E)$, we delete its children u where $\nu(E(u))$ is false, and their descendants. Then, independently for each mux node u labeled $\mathsf{mux}(\mathsf{Pr})$, we select one of its children u' according to the corresponding probability distribution Pr and delete the other children and their descendants, the probability of the choice is $\mathsf{Pr}(u')$. We do not delete any of the children of det nodes.
3. We then remove in turn each distributional node, connecting each ordinary child u of a deleted distributional node with its lowest ordinary ancestor u'.

The result of this third step is a random document $\mathscr{P}$. The probability of a random generation is defined by the probabilities of the choices done during the generation and it is the product of p_ν, the probability of the variable assignment we chose in the first step, with all $\mathsf{Pr}(u')$, the probabilities of the choices made in the second step for the mux nodes. The probability $\mathsf{Pr}(\mathscr{P})$ is the sum of the probabilities across all possible random generations that yield $\mathscr{P}$.

Example 2 *One can obtain the document d_{MBOX} in Figure 5 by applying the randomized process to either of the p-documents $\widehat{\mathscr{P}}_{MBOX\text{-}L}$ and $\widehat{\mathscr{P}}_{MBOX\text{-}G}$.*

For $\widehat{\mathscr{P}}_{MBOX\text{-}L}$ in Figure 2 this can be done by making three choices, namely, (i) the right branch of the node n_{70} with probability 0.65, (ii) the right branch of n_{72} with probability 0.8, and finally (iii) the left branch of the node n_{73} with probability 0.9. Then the probability of d_{MBOX} is the product of the probabilities of the choices $\mathsf{Pr}(d) = 0.65 \times 0.8 \times 0.9 = 0.468$.

For $\widehat{\mathscr{P}}_{MBOX\text{-}G}$ in Figure 4 this can be done by assigning $\nu = \{x/\mathsf{false}, z/\mathsf{false}\}$. According to ν the nodes n_{21}, n_{30}, and n_{45}, that are children of cie *distributional nodes, are to be deleted from $\widehat{\mathscr{P}}_{MBOX\text{-}G}$ and do not occur in the resulting document, since the edges incoming to these nodes are labeled with formulas evaluated by ν to false. The other children of the* cie *distributional nodes remain in the resulting document, since the edges incoming to these nodes are labeled with formulas evaluated by ν to true. This is the only valuation leading to the result document d_{MBOX}, thus the probability of d_{MBOX} is the product of the probabilities of the assignments for each variable, that is, of $\mathsf{Pr}(\neg x) = 0.85$ and $\mathsf{Pr}(\neg z) = 0.8$, and it is equal to $\mathsf{Pr}(d) = 0.85 \times 0.8 = 0.68$.* ■

$\mathsf{PrXML}^{\mathsf{cie}}$ vs $\mathsf{PrXML}^{\mathsf{mux,det}}$. $\mathsf{PrXML}^{\mathsf{cie}}$ and $\mathsf{PrXML}^{\mathsf{mux,det}}$ are both capable of representing any discrete px-space. However, they correspond to different trade-offs between succinctness and query efficiency. As shown in (Abiteboul et al., 2009), $\mathsf{PrXML}^{\mathsf{cie}}$ is an exponentially more compact representation system than $\mathsf{PrXML}^{\mathsf{mux,det}}$, which itself is as compact as $\mathsf{PrXML}^{\mathsf{mux,ind}}$. On the other hand, Kimelfeld et al. (2008, 2009) have shown that there existed a polynomial-time algorithm (in data complexity) for computing the probability of all *tree-pattern queries* over $\mathsf{PrXML}^{\mathsf{mux,det}}$, while all non-trivial queries are #P-complete over $\mathsf{PrXML}^{\mathsf{cie}}$.

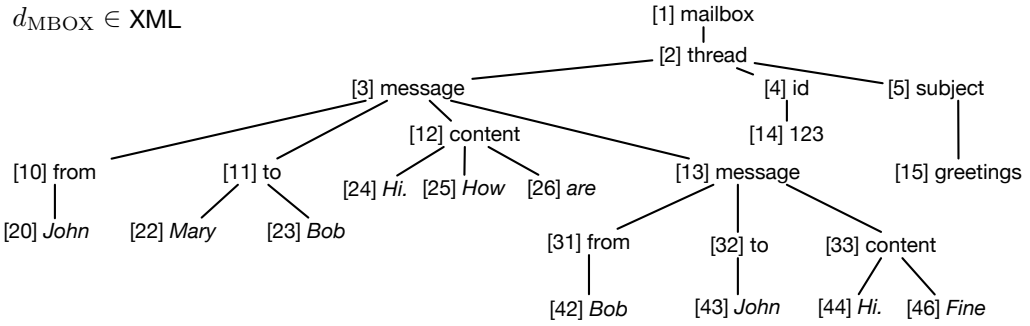

Figure 5: XML document d_{MBOX}: Mailbox organized in threads

3.2 Continuous Probabilistic XML

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 an uncountably infinite set of trees with a continuous distribution, which assigns probabilities to (typically infinite) sets of trees, the possible events. In order to support such a model we now consider only documents whose leaves are labeled with real numbers. Let $\mathscr{D}$ be the set of all documents where the values of the leaves range over the reals. Then, a *continuous* px-*space* is a continuous distribution over $\mathscr{D}$. We refer to a textbook on measure and probability theory such as (Ash & Doléans-Dade, 2000) for the definitions of the concepts used in this section.

To support (possibly 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 $\mathsf{cont}(D)$, 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.

Example 3 *Consider the probabilistic XML document $\widehat{\mathscr{P}}_{CONS}$ in Figure 6 that belongs to* $\mathsf{PrXML}^{\mathsf{cont,mux}}$*, since it has two types of distributional nodes, namely,* cont *and* mux*. Documents represented by this p-document follow the DTD in Figure 3 on the right.*

The p-document collects results of electricity-consumption monitoring by sensors installed in two rooms: room 1 and room 2. The data in the first room is measured during September 3 and 4. The consumption on September 3 is reported to be 15 *units. This measurement is imprecise, but we know (from the sensor's manufacturer) that the error is normally distributed around the reported value with variance* 3*. We represent this fact by a continuous node n_{25} labeled* $\mathsf{cont}(N(15,3))$*. On September 4 there is a communication problem with this sensor and the reported consumption is either* 50 *units with probability* 0.1*, or* 52 *units with probability* 0.9*. Moreover when the consumption is higher than* 30 *the variance of the measurement imprecision is getting higher and equal to* 5*. This data is represented using the* mux *distributional node n_{27} with two children that are labeled with continuous distributions* $\mathsf{cont}(N(50,5))$ *and* $\mathsf{cont}(N(52,5))$*.*

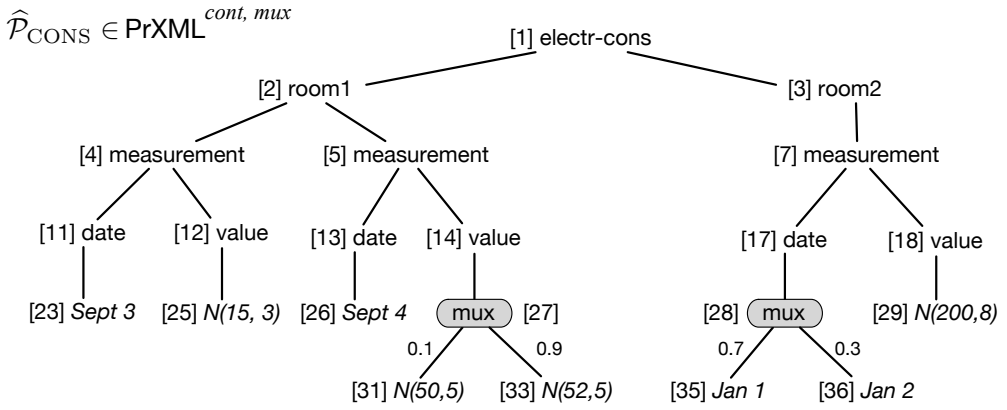

Figure 6: $\text{PrXML}^{\text{mux,cont}}$ p-document $\widehat{\mathscr{P}}_{\text{CONS}}$: Electricity consumption

In the second room we have uncertainty on the date when the measurement stored in the node n_{29} was collected. ■

Any finitely representable distribution can appear in a cont node: uniform distributions, piecewise polynomials, Poisson distributions, etc. We consider in more detail the example of *normal* or *Gaussian distributions*, that were used in the example above.

The probability density function of a Gaussian distribution $N(\mu, \sigma^2)$ of mean μ and variance σ^2 is:

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \mathrm{e}^{-(x-\mu)^2/(2\sigma^2)}.$$

The density function has a shape of a bell centered in the mean value μ with variance σ^2. The Gaussian distribution is often used to describe, at least approximately, measurements that tends to cluster around the mean. For example, if a sensor indicates a temperature of 30, then the real value might be slightly different from 30, but not too much. The further the value is from 30, the lower the chance that this is the actual temperature. This behavior can be nicely captured by a Gaussian distribution with mean of 30 and a small variance.

We define the semantics $[\![\widehat{\mathscr{P}}]\!]$ of continuous p-documents of $\text{PrXML}^{\text{cont,cie,mux,det}}$ as a continuous px-space described by a probability distribution function $\text{Pr}_{\widehat{\mathscr{P}}}$. More precisely, the semantics is defined in two steps (Abiteboul et al., 2010).

1. Let $\widehat{\mathscr{P}} \in \text{PrXML}^{\text{cont,cie,mux,det}}$ and $\widehat{\mathscr{P}'} \in \text{PrXML}^{\text{cie,mux,det}}$ be the p-document obtained from $\widehat{\mathscr{P}}$ by replacing every continuous node with an arbitrary value, say, 0. $[\![\widehat{\mathscr{P}'}]\!]$ is a (discrete) px-space $(\{(d_1, p_1) \ldots (d_n, p_n)\})$ with $\sum p_i = 1$. For a given $1 \leq i \leq n$, we consider the document $\widehat{\mathscr{P}_i}$ of $\text{PrXML}^{\text{cont}}$ obtained by putting back in d_i the continuous nodes of $\widehat{\mathscr{P}}$, where the corresponding leaves still exist.
2. Let $D_{i_1} \ldots D_{i_k}$ be the k probability distributions over the real numbers represented in the cont nodes of $\widehat{\mathscr{P}_i}$. We define then a continuous probability distribution Pr_i over $\mathbb{R}^k$ as the product distribution (Ash & Doléans-Dade, 2000) of the D_{ij}'s, i.e., the unique distribution such that $\text{Pr}_i(X_1 \times \cdots \times X_k) = D_{i_1}(X_1) \times \cdots \times D_{i_k}(X_k)$. Let $\mathscr{D}_i \subseteq \mathscr{D}$ be all the documents

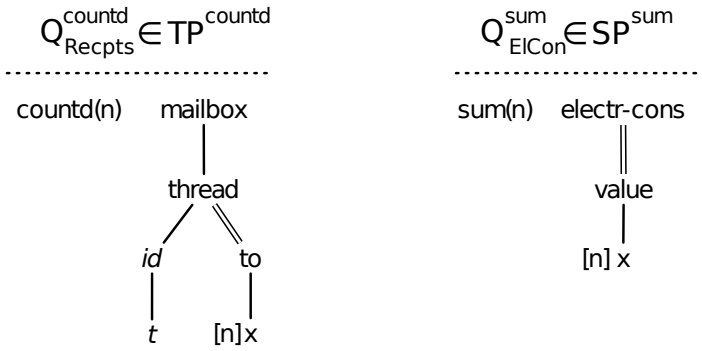

Figure 7: Two types of queries: TP query $Q^{\mathsf{countd}}_{\mathrm{Recpts}}$ and SP query $Q^{\mathsf{sum}}_{\mathrm{ElCon}}$

that have the same structure as d_i, that is, that may differ from d_i on the values of the leaves only. Then, the probability of every subset $\mathscr{D}' \subseteq \mathscr{D}$ defined by $\Pr_{\widehat{\mathscr{P}}}$ is

$$\Pr_{\widehat{\mathscr{P}}}(\mathscr{D}') := \sum_{i=1}^{n} p_i \cdot \Pr_i(\mathscr{D}' \cap \mathscr{D}_i).$$

4 Querying Probabilistic XML

We explain here how p-documents can be queried, and what the complexity of this operation is, depending on the query language considered. Most of the results from this section are from (Kimelfeld et al., 2009; Abiteboul et al., 2010). They will be useful in understanding the kind of mining tasks that are tractable, as detailed in the next section. We start by introducing standard tree-pattern queries, before moving to aggregate queries, which are of special interest in mining tasks that involve data values. We first define how these queries apply to regular documents, and then explain how to query probabilistic documents.

Tree-pattern queries. A *tree pattern*, denoted Q, is a tree with two types of edges: child edges and descendant edges. The nodes of the tree are labeled by a labeling function with either labels or variables (e.g., x, y, z), such that no variable occurs more than once, that is, *join variables* are not allowed.

A *tree-pattern query* has the form $Q[\bar{n}]$, where Q is a tree pattern and $\bar{n}$ is a tuple of nodes of Q (defining its output). We sometimes identify the query with the pattern and simply write Q instead of $Q[\bar{n}]$ if the tuple $\bar{n}$ is not important or clear from the context. If $\bar{n}$ is the empty tuple, we say that the query is Boolean. If the set of edges of a query is a linear order, the query is called a *single-path query.* We denote the set of all tree-pattern queries as TP, and the subclass of single-path queries as SP.

Example 4 *Consider the two queries in Figure 7, ignoring for now the aggregate functions* countd *and* sum. *The left query is over the mailbox DTD and the right one over the electricity*

consumption DTD, both can be found in Figure 3. Descendant edges are marked with double lines.

The left query is a TP *query since it has branching. It asks for people that received messages inside a given thread t. The right query is in* SP *since it has no branching. It returns all electricity consumption values recorded in the document.* ■

A *valuation* maps query nodes to document nodes. A document *satisfies* a 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 and the labeling. That is,

1. nodes connected by child/descendant edges are mapped to nodes that are children/descendants of each other;
2. query nodes with label a are mapped to document nodes with label a.

We define that applying a query $Q[\bar{n}]$ to a document d returns a set of tuples of nodes:

$$Q(d) := \{\nu(\bar{n}) \mid \nu \text{ satisfying valuation for } Q\}.$$

Aggregate Functions. An *aggregate function* maps a finite bag of values (e.g., rationals) into some domain (possibly the same or different). In particular, we assume that any aggregate function is defined on the empty bag. We consider here standard aggregate functions: sum, count, min, countd (count distinct), and avg (average) under their usual semantics. Our results easily extend to max and top$k$.

Aggregate functions can be naturally extended to work on documents d: the result $\alpha(d)$ is $\alpha(B)$ where

$$B = \{\!\{l_1, \ldots, l_n\}\!\}$$

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, e.g., rational numbers for sum. Again to simplify, we ignore this issue here and assume they all have the proper type. It is straightforward to extend this model and the corresponding results with a more refined treatment of typing.

A class of aggregate functions that play an important role in our investigation are *monoid* ones (Cohen, Nutt, & Sagiv, 2006), because they can be handled by a divide-and-conquer strategy. It turns out that sum, count, min, max, and top$k$ are monoid aggregate functions. For example, sum can be handled as

$$\mathsf{sum}(\{\!\{l_1, \ldots, l_n\}\!\}) = \mathsf{sum}(\{\!\{l_1\}\!\}) + \cdots + \mathsf{sum}(\{\!\{l_n\}\!\}) = l_1 + \cdots + l_n.$$

On the other hand, it is easy to check that neither avg nor countd are monoid aggregate functions and, as we see further, in general it is more difficult to handle them.

Aggregate Queries over Documents. An *aggregate* TP-query has the form $Q[\alpha(n)]$, where Q is a tree pattern, n is a node of Q, and α is an aggregate function. The semantics of such an aggregate query $Q[\alpha(n)]$ is given by this three-step evaluation:

1. Evaluate the non-aggregate query $Q' := Q[n]$ over d and obtain a set of nodes $Q'(d)$.

2. Compute the *bag* B of labels of $Q'(d)$, that is

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

where θ is the labeling function of d.

3. Finally, apply α to B.

We denote the value resulting from evaluating an aggregate query Q over d as $Q(d)$.

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

Example 5 *Consider the two aggregate queries in Figure 7, where the aggregation nodes are marked with* $[n]$*. The query* $Q^{\mathsf{countd}}_{Recpts} \in \mathsf{TP}^{\mathsf{countd}}$ *asks for the number of different persons receiving messages inside a given thread* t*. The evaluation of this query over the document* d_{MBOX} *from Figure 5 with* $t = 123$ *returns the value* 3*, since there are three distinct persons concerned, namely, Mary, Bob and John. The query* $Q^{\mathsf{sum}}_{ElCon} \in \mathsf{SP}^{\mathsf{sum}}$ *returns the total electricity consumption across all rooms.* ■

The syntax and semantics above can be generalized in a straightforward fashion to aggregate queries with an SQL-like `GROUP BY` operator.

Queries over Probabilistic Documents. We have defined the semantics of queries over regular documents. We now define their semantics over a p-document $\widehat{\mathscr{P}}$ using the possible-world semantics of p-documents.

First, consider a Boolean query Q. The semantics of Q over $\widehat{\mathscr{P}}$, denoted $Q(\widehat{\mathscr{P}})$ is defined as the probability that Q evaluates to true in the possible-world semantics of $\widehat{\mathscr{P}}$:

$$Q(\widehat{\mathscr{P}}) = \Pr(Q(d) \mid d \in [\![\widehat{\mathscr{P}}]\!]).$$

Similarly, since an aggregate query Q^α maps elements of the probability space $[\![\widehat{\mathscr{P}}]\!]$ to values in the range of α, we can see $Q^\alpha(\widehat{\mathscr{P}})$ as a random variable. We therefore define the result of applying Q^α to $\widehat{\mathscr{P}}$ as the distribution of this random variable,

$$(Q^\alpha(\widehat{\mathscr{P}}))(c) = \mathsf{Pr}(Q^\alpha(d) = c \mid d \in [\![\widehat{\mathscr{P}}]\!]),$$

for c in the range of α. Note that if the non-aggregate part Q does not match a document d, then we define $Q^\alpha(d)$ depending on the type of α. If α is a monoid function, we define $Q^\alpha(d)$ as the neutral element corresponding to α, e.g., 0 for sum, $-\infty$ for max. If α is countd or avg, we set $Q^\alpha(d) = 0$.

Observe that for the case of continuous p-documents the probability defining $(Q^\alpha(\widehat{\mathscr{P}}))(c)$ is equal to zero, unless there are Dirac functions labeling the leaves of $\widehat{\mathscr{P}}$. A more appropriate definition for continuous p-documents would be on an interval, rather than a single point, that is,

$$(Q^\alpha(\widehat{\mathscr{P}}))(c_1, c_2) = \mathsf{Pr}(Q^\alpha(d) \in (c_1, c_2) \mid d \in [\![\widehat{\mathscr{P}}]\!]),$$

for c_1 and c_2 in the range of α.

We use the notation $Q(\mathscr{P})$ or $Q^\alpha(\mathscr{P})$ to denote the random variable corresponding to the application of query Q or Q^α over the random document $\mathscr{P}$ associated to a p-document $\widehat{\mathscr{P}}$.

Example 6 *The evaluation of the query* $Q^{\mathsf{countd}}_{Recpts}$ *over the* mux-det *p-document* $\widehat{\mathscr{P}}_{MBOX\text{-}L}$ *gives the distribution:*

$$Q^{\mathsf{countd}}_{Recpts}(\widehat{\mathscr{P}}_{MBOX\text{-}L}) = \{(1,0.35),\,(3,0.65)\}\,.$$

Indeed, $\widehat{\mathscr{P}}_{MBOX\text{-}L}$ *has three* mux *distributional nodes* n_{70}*,* n_{72} *and* n_{73} *with two children each, therefore, there are eight possible worlds for this p-document. It is easy to see that in this example all generations yield different documents. Observe that the document* d_{MBOX} *can be generated by choosing the right branch of* n_{70}*, the right branch of* n_{72}*, and the left branch of* n_{73}*. Considering these distributional nodes as the vector* (n_{70},n_{72},n_{73})*, the eight possible worlds are* $w_1 = (l,l,l)$*,* $w_2 = (r,l,l)$*,* $w_3 = (l,r,l)$*,* $w_4 = (l,l,r)$*,* $w_5 = (r,r,l)$*,* $w_6 = (r,l,r)$*,* $w_7 = (l,l,r)$*,* $w_8 = (r,r,r)$*, where l (resp. r) stands for the choice of the left (resp. right) child of the corresponding* mux *node. Note that only the choice in node* n_{70} *has an influence over the result of the aggregate query* $Q^{\mathsf{countd}}_{Recpts}$*. One can see that the probabilities of the worlds are, for example,* $\mathsf{Pr}(w_1) = 0.35 \times 0.2 \times 0.9$ *and* $\mathsf{Pr}(w_8) = 0.65 \times 0.8 \times 0.1$*.*

The probability that there are three distinct recipients is:

$$\mathsf{Pr}(Q^{\mathsf{countd}}_{Recpts}(\mathscr{P}_{MBOX\text{-}L}) = 3) = \mathsf{Pr}(w_2) + \mathsf{Pr}(w_5) + \mathsf{Pr}(w_6) + \mathsf{Pr}(w_8) = 0.65,$$

and otherwise, there is only one recipient (John).

Similarly, one can check that the evaluation of the query $Q^{\mathsf{countd}}_{Recpts}$ *over the* $\mathsf{PrXML}^{\mathsf{cie}}$ *p-document* $\widehat{\mathscr{P}}_{MBOX\text{-}G}$ *gives the distribution:*

$$Q^{\mathsf{countd}}_{Recpts}(\widehat{\mathscr{P}}_{MBOX\text{-}G}) = \{(1,0.2),\,(2,0.12),\,(3,0.68)\}$$

by considering all possible valuations of the variables x and z. ∎

Complexity of Querying and Aggregating p-Documents We now summarize complexity results for both querying and aggregating probabilistic XML, from (Kimelfeld et al., 2009; Abiteboul et al., 2010). We are interested in *data complexity* (Vardi, 1982), i.e., the complexity is measured with respect to the size of the input document, the query being fixed. To simplify, we do not consider continuous distributions here, see (Abiteboul et al., 2010) for details. We are interested in three computational problems.

The first problem is the computation of the probability

$$\mathsf{Pr}(\mathscr{P} \models Q)$$

that a given Boolean query Q matches a random document $\mathscr{P}$ of a given probabilistic document $\widehat{\mathscr{P}}$. The second problem is the computation of the probability

$$\mathsf{Pr}(Q^{\alpha}(\mathscr{P}) = c)$$

that the value of a given aggregate query Q^{α} over a random document $\mathscr{P}$ of a given probabilistic document $\widehat{\mathscr{P}}$ is equal to a given value c. The third problem is the computation of moments

$$\mathsf{E}^{k}(Q^{\alpha}(\mathscr{P}))$$

Table 1: Worst-case data complexity of computation problems for $\mathsf{PrXML}^{\mathsf{mux,det}}$: the probability of a query match $\mathsf{Pr}(\mathscr{P} \models Q)$ for a Boolean query Q; the probability of an aggregate value $\mathsf{Pr}(Q^\alpha(\mathscr{P}) = c)$ and the moment $\mathsf{E}^k(Q^\alpha(\mathscr{P}))$ of degree k for an aggregate query Q^α where $\alpha \in \{\mathsf{count}, \mathsf{sum}, \mathsf{min}, \mathsf{countd}, \mathsf{avg}\}$.

for $\mathsf{PrXML}^{\mathsf{mux,det}}$	SP, TP	
$\mathsf{Pr}(\mathscr{P} \models Q)$	PTIME	
$\mathsf{Pr}(Q^\alpha(\mathscr{P}) = c)$	$\mathsf{count}, \mathsf{min}$	in PTIME
	$\mathsf{sum}, \mathsf{avg}, \mathsf{countd}$	$\mathsf{FP}^{\#\mathsf{P}}$-complete

	SP	TP	
$\mathsf{E}^k(Q^\alpha(\mathscr{P}))$	in PTIME	avg	in $\mathsf{FP}^{\#\mathsf{P}}$
		others	in PTIME

of degree k for a given aggregate query Q^α and a natural number k (moments are useful summaries of probabilistic distributions, e.g., the moment of degree 1 is the expected value).

For the $\mathsf{PrXML}^{\mathsf{cie}}$ model all problems are intractable, namely $\mathsf{FP}^{\#\mathsf{P}}$-complete ($\mathsf{FP}^{\#\mathsf{P}}$ is a class of computational problems that are defined using counting counterparts of NP problems) with the exception of the computation of moments for $\mathsf{SP}^{\mathsf{sum}}$ and $\mathsf{SP}^{\mathsf{count}}$, which is polynomial-time.

For $\mathsf{PrXML}^{\mathsf{mux,det}}$, the situation is more involved and is described in Table 1. The probability of a query match can be computed in polynomial time for TP. Note that for relational databases, computational problems for aggregate queries are usually more difficult than for non-aggregate queries (Cohen et al., 2006). This situation also holds for $\mathsf{PrXML}^{\mathsf{mux,det}}$ when considering the complexity of a probability of an aggregate value for $\mathsf{TP}^{\mathsf{countd}}$ or $\mathsf{TP}^{\mathsf{avg}}$ queries, that is, for queries with non-monoid aggregate functions. For $\mathsf{TP}^{\mathsf{count}}$ or $\mathsf{TP}^{\mathsf{min}}$ the complexity of aggregate value probability computation is the same as for the query match problem. Moment computation is tractable for SP^α with every considered α and even for TP^α when $\alpha \in \{\mathsf{count}, \mathsf{sum}, \mathsf{min}, \mathsf{countd}\}$.

System Issues. Most existing work on probabilistic XML is at the theoretical level. At the time of this writing, an actual full-fledged system that supports the management of probabilistic XML is still missing. There are, however, several attempts at meeting this need. A first approach relies on encoding XML data into relations (Hollander & Keulen, 2010): a probabilistic XML database thus becomes encoded as a probabilistic relational database, that can then be managed using a system such as MayBMS, for which efficient query evaluation algorithms have been developed, both exact and approximate. The downside of the approach is that relational databases do not exploit the specific characteristics of tree-like data encoded into databases, that for instance makes tree-pattern queries over local models tractable. Another direction is to natively process tree-pattern queries over XML data: this is what is done in the implementation of the EvalDP algorithm (Kimelfeld et al., 2008, 2009) for $\mathsf{PrXML}^{\mathsf{mux,det}}$ and this is also the idea behind ProApproX (Senellart & Souihli, 2010b, 2010a), which implements approximate query answering using various schemes of sampling methods (Kimelfeld et al., 2009), on top of a native

XML query processor. Either approach has ignored the specific aspects of indexing and storage of probabilistic XML. An actual system would need to reduce the number of disk and memory accesses to data and probabilistic annotations. A first observation is that it is sometimes more efficient to store the marginal probabilities of nodes in a document rather than the probabilities of nodes conditioned by the existence of their parent (Li, Shao, & Chen, 2006).

5 Mining Information from Probabilistic XML

In this section we show how techniques developed for query answering over probabilistic XML can be applied to data mining tasks. We illustrate the techniques on a number of use cases, for our two example scenarios: a collection of emails (such as the one from $\widehat{\mathscr{P}}_{\text{MBOX-G}}$ or $\widehat{\mathscr{P}}_{\text{MBOX-L}}$) and data collected from sensors (e.g., $\widehat{\mathscr{P}}_{\text{CONS}}$).

Mining Frequent Items. Frequent itemsets play an essential role in many data mining tasks that look for interesting patterns in databases, such as association rules, correlations, sequences, episodes, classifiers, clusters. Association rule mining, especially, is one of the most common data mining task, originally motivated by *market basket* analysis. This problem assumes that a number of items, like bread or milk, are bought by customers who fill their market baskets with (subsets of) these items. The task is to learn which items people usually buy together, this information being valuable to position items on market shelves and influence the way typical customers browse stores. Mining frequent itemsets is used as the basis of association rules mining since they guarantee that the rules are based on sets of items with a high *support* (Tan, Steinbach, & Kumar, 2005).

In our mailbox example scenario a traveling agency might want to find cities that are often mentioned in messages and then use this information for targeted advertisement. The following request checks for frequent terms in the mailbox repository:

> Find a word that with probability higher than θ occurs more than k times in emails across all the threads.

where θ is the *quality* of frequent item prediction.

One possible strategy for running this request is to check for each city which is operated by the agency whether this city is frequently discussed in the mailbox repository. That is, for a given word w and a p-document $\widehat{\mathscr{P}}$ this request could be formulated as the decision problem $\mathsf{Pr}(Q^{\mathsf{count}}(\mathscr{P}) \geq k) \geq \theta$ for the following $\mathsf{SP}^{\mathsf{count}}$ aggregate query written in XPath notation:

$$Q^{\mathsf{count}} = \mathsf{count}(//\text{content}/w). \tag{1}$$

For every p-document $\widehat{\mathscr{P}}$ the range of the random variable $Q^{\mathsf{count}}(\mathscr{P})$ (the number of different values with non-zero probability) is bounded by the number of leaves in $\widehat{\mathscr{P}}$, that is, by the size $|\widehat{\mathscr{P}}|$ of $\widehat{\mathscr{P}}$. Therefore, taking into account that the probabilistic events $Q^{\mathsf{count}}(\mathscr{P}) = n_1$ and $Q^{\mathsf{count}}(\mathscr{P}) = n_2$, where $n_1 \neq n_2$, are disjoint, we obtain

$$\mathsf{Pr}(Q^{\mathsf{count}}(\mathscr{P}) \geq k) = \sum_{n=k}^{|\widehat{\mathscr{P}}|} \mathsf{Pr}(Q^{\mathsf{count}}(\mathscr{P}) = n).$$

This sum can be computed in polynomial time as far as the computation of every component is polynomial. In the previous section we discussed that the tractability of probability computation $\mathsf{Pr}(Q^{\mathsf{count}}(\mathscr{P}) = n)$ for a given aggregate value depends on the type of the p-document $\widehat{\mathscr{P}}$. For $\widehat{\mathscr{P}} \in \mathsf{PrXML}^{\mathsf{cie}}$ the computation is hard, but for $\widehat{\mathscr{P}} \in \mathsf{PrXML}^{\mathsf{mux,det}}$ the computation is indeed tractable for $\mathsf{TP}^{\mathsf{count}}$.

In the case when $Q \in \mathsf{TP}$, the computation of $\mathsf{Pr}(Q^{\mathsf{count}}(\mathscr{P}) = n)$ can be done using the techniques of (Cohen et al., 2008) that are based on dynamic programming and boil down to computing the probability of a query match under aggregation constraints. In the case when $Q \in \mathsf{SP}$, the computation of $\mathsf{Pr}(Q^{\mathsf{count}}(\mathscr{P}) = n)$ can be done using techniques from (Abiteboul et al., 2010), which allow to compute the entire distribution $Q^{\mathsf{count}}(\widehat{\mathscr{P}})$ in one bottom-up scan of $\widehat{\mathscr{P}}$, computing thus convex sums and convolutions of distributions.

Example 7 *We now compute the probability* $\mathsf{Pr}(Q^{\mathsf{count}}(\mathscr{P}_{\textit{MBOX-L}}) \geq 2)$ *for the query of* (1) *and* $w =$ *"Hi". One can see that the probability* $\mathsf{Pr}(Q^{\mathsf{count}}(\mathscr{P}_{\textit{MBOX-L}}) = 1)$ *is the same as the probability to choose the right child of the node* n_{73}*, that is, equal to* 0.1*. The probability of* $\mathsf{Pr}(Q^{\mathsf{count}}(\mathscr{P}_{\textit{MBOX-L}}) = 2)$ *is equal to the probability of the left-side child, that is, to* 0.9*. Moreover, for* $n > 2$*,* $\mathsf{Pr}(Q^{\mathsf{count}}(\mathscr{P}_{\textit{MBOX-L}}) = n) = 0$*. Therefore,*

$$\mathsf{Pr}(Q^{\mathsf{count}}(\mathscr{P}_{\textit{MBOX-L}}) \geq 2) = \mathsf{Pr}(Q^{\mathsf{count}}(\mathscr{P}_{\textit{MBOX-L}}) = 2) = 0.9$$

Assuming the required quality of frequent item prediction θ *is* 0.8*, then the word "Hi" is a frequent item, since it has high frequency has the probability* $0.9 > 0.8$*.* ■

Mining Co-Occurring Items. Another important application of frequent itemset mining is the extraction of social-network graphs from communication traces. These graphs are used, for example, in recommender systems or social search applications, when one wants to use the social network of an individual to personalize recommendations or search results. Mining a social-network graph in our mailbox setting can be carried out in the following way:

> Find all pairs of people that with probability higher than θ are emailing in the same thread.

Every such pair can be interpreted as an edge of the social network. We could go further and also require that two persons have exchanged a minimum number of messages to categorize them as "friends", for instance, but let us consider this simple request for now.

One possible strategy for running this request is to check for each pair of names (w_1, w_2) mentioned in a p-document whether their probability of co-occurrence is high enough. That is, for a p-document $\widehat{\mathscr{P}}$ this request can be formulated as the decision problem $\mathsf{Pr}(\mathscr{P} \models Q) \geq \theta$ for the Boolean TP query:

$$Q = //\text{thread}[.//\text{from} = w_1 \text{ and } .//\text{from} = w_2]. \tag{2}$$

The probability of a Boolean query match for TP queries can be computed efficiently over $\mathsf{PrXML}^{\mathsf{mux,det}}$ p-documents using the techniques of (Kimelfeld et al., 2008). On the other hand, the computation is intractable over $\mathsf{PrXML}^{\mathsf{cie}}$ p-documents.

Example 8 *We now compute the probability* $\mathsf{Pr}(\mathscr{P} \models Q) \geq \theta$ *for the query of* (2)*, persons* $w_1 =$ *"John",* $w_2 =$ *"Mary" and threshold* $\theta = 0.7$*. The probability* $\mathsf{Pr}(\mathscr{P} \models Q)$ *is the same as the probability to choose the left child of node* n_{72} *and thus equal to* 0.2*, which is less than the threshold. We conclude that John and Mary are not friends. Besides, if* $w_2 =$ *"Bob", then* $\mathsf{Pr}(\mathscr{P} \models Q) = 0.8$ *and since* $0.8 > 0.7$ *we conclude that John and Bob are friends.* ■

Mining Popular Items. In applications such as recommender systems an important task is to find popular items (Ricci, Rokach, Shapira, & Kantor, 2010). A classic example of a recommender system is Amazon, where products like books or electronics are recommended to people. These recommendations are based on both the popularity of items and the purchase history of individual customers. Popular items are products that either have high users' ratings or have been chosen by many customers. In the mailbox example, an item could be a thread and its popularity could be measured by the number of people that received messages sent within the thread. An example of a request for popular threads is the following.

> Find all threads that with probability higher than θ attract more than k different recipients.

This request can be formulated as an aggregate query. For a given thread identifier t and a p-document $\widehat{\mathscr{P}}$, decide whether $\mathsf{Pr}(Q^{\mathsf{countd}}(\mathscr{P}) \geq k) \geq \theta$ holds where Q^{countd} is the aggregate query:

$$Q^{\mathsf{countd}} = \mathsf{countd}(//\mathrm{thread}[\mathrm{id} = t]//\mathrm{to}). \tag{3}$$

As in the case of $Q^{\mathsf{count}}(\mathscr{P})$, the range of the random variable $Q^{\mathsf{countd}}(\mathscr{P})$ is limited by the number of leaves in $\widehat{\mathscr{P}}$ and we can reformulate this query as the sum:

$$\mathsf{Pr}(Q^{\mathsf{countd}}(\mathscr{P}) \geq k) = \sum_{n=k}^{|\widehat{\mathscr{P}}|} \mathsf{Pr}(Q^{\mathsf{countd}}(\mathscr{P}) = n).$$

We are again in the case where the tractability of the mining task depends on the tractability of the probability computation $\mathsf{Pr}(Q^{\mathsf{countd}}(\mathscr{P}) = n)$. This time the aggregate function is countd, that is, a non-monoid function, and, consequently, the probability computation is intractable in both models $\mathsf{PrXML}^{\mathsf{cie}}$ and $\mathsf{PrXML}^{\mathsf{mux,det}}$, even for $\mathsf{SP}^{\mathsf{countd}}$ queries. This means that the only known way to compute the probability is to construct the whole probability space of p-documents $[\![\widehat{\mathscr{P}}]\!]$ represented by $\widehat{\mathscr{P}}$, then to evaluate the aggregate query Q^{countd} in each document of $[\![\widehat{\mathscr{P}}]\!]$ separately and, finally, to sum the probabilities of the documents $d \in [\![\widehat{\mathscr{P}}]\!]$ where $Q^{\mathsf{countd}}(d) = n$.

Example 9 *We compute the probability* $\mathsf{Pr}(Q^{\mathsf{countd}}(\mathscr{P}_{MBOX\text{-}G}) \geq 3)$ *for the query of* (3) *and* $t =$ *"greetings". As discussed in Example 6 there are four documents in* $[\![\widehat{\mathscr{P}}_{MBOX\text{-}G}]\!]$ *corresponding to different assignments of the variables x and z. One can see that the only world where there are at least three different people subscribed to the thread "greetings" is when both x and z are assigned to false. The three recipient in this case are Mary, Bob, (who received the first message) and John (who received the second message). The probability of such a world is* $0.85 \times 0.8 = 0.68$ *Therefore,*

$$\mathsf{Pr}(Q^{\mathsf{countd}}(\mathscr{P}_{MBOX\text{-}G}) \geq 3) = 0.68$$

Assume that the required quality of popular item prediction θ is again 0.8*. Then the thread "greetings" is not a popular item, since its popularity has lower probability.* ■

Mining Continuous Data. One application that motivated the introduction of a continuous model for probabilistic data is monitoring in sensor networks (Cheng, Kalashnikov, & Prabhakar, 2003; Abiteboul et al., 2010). A typical example of sensor network is sensors installed in a river to measure either the temperature, or the level of water, or the concentration of different chemical components to predict floods of poisoning of the river. Another example is sensors that measure consumption of, say, electricity or Internet bandwidth. In sensor networks, monitoring of measurements' behavior is the main task. One wants to be alerted whenever the level of water in the river is growing too fast, or when the electricity consumption is higher than expected. In this case one can make actions to avoid flooding or to repair, substitute the equipment that is wasting the electricity.

Coming back to the continuous p-document $\widehat{\mathscr{P}}_{\text{CONS}}$ we would like to perform the following monitoring task:

> Find all rooms where the expected value of the total electricity consumption is more than v units.

For a given room r, p-document $\widehat{\mathscr{P}}$, and an expected electricity consumption of v units this request cam be reformulated as the decision problem $\mathsf{E}(Q^{\mathsf{sum}}(\mathscr{P})) \geq v$ for the aggregate query:

$$Q^{\mathsf{sum}} = \mathsf{sum}(/*/r/*/\text{value}). \tag{4}$$

The expected value $\mathsf{E}(Q^{\mathsf{sum}}(\mathscr{P}))$ can be computed in polynomial time for continuous p-documents from $\mathsf{PrXML}^{\mathsf{cont,mux,det}}$ as long as one is able to compute the expected values of the continuous distributions stored on the leaves of p-documents (Abiteboul et al., 2010). The computation proceeds exactly in the same way as for discrete documents.

Example 10 *Let us compute the expected value* $\mathsf{E}(Q^{\mathsf{sum}}(\mathscr{P}))$ *for the query of* (4)*, room* $r =$ *"room1" and threshold* $v = 150$*. This can be done by separately computing the expected values of the sum in the left and in the right child of* n_{27}*, and then by combining the values as the convex sum with the corresponding coefficients* 0.1 *and* 0.9*, see (Abiteboul et al., 2010) for details. The expected value of a Gaussian is obviously its mean. The result is*

$$\mathsf{E}(Q^{\mathsf{sum}}(\mathscr{P})) = 15 + (0.1 \times 50 + 0.9 \times 52) = 66.8.$$

Since the expected value is below the threshold, we conclude that the electricity consumption in room 1 is fine. ■

6 Conclusion and Future Research

Conclusion. As shown in the previous section, a number of data mining tasks are indeed possible over probabilistic XML data, and it is possible to use probabilistic characterizations

(probability higher than a threshold, expected value, etc.) to define these tasks. The efficiency of the proposed methods critically depends on the kind of data model considered (especially, global vs local models) and on the tractability of querying over these models, for a query language (Boolean vs aggregate queries, single-path vs tree-pattern) that depends of the task considered.

Future Research. The foundations of probabilistic XML are now quite well established. The connection between local and global models is well-understood, and the complexity of querying these models with tree-pattern queries has been investigated in length. Challenges in modeling and querying lie in more expressive models (such as the recursive Markov chains of (Benedikt et al., 2010)), more expressive query languages (especially, involving value joins, cf. (Senellart & Abiteboul, 2007; Abiteboul et al., 2010)), and understanding better how results from the relational and XML settings relate to each other. An important missing aspect is the building of an actual system that leverage the knowledge brought by theoretical studies of probabilistic XML. We have already pointed at several works in that direction.

On the mining side, however, much remains to be done. The approaches for mining probabilistic XML data proposed in this chapter directly rely on a reformulation of the problem as a query or a set of queries. In some cases, this might not be the most efficient technique. Consider for instance the problem of discovering frequent itemsets. Obviously, an approach that would consider all possible itemsets of a bounded size and evaluate a query for each of these would be impractical on large data even though this may amount to running a polynomial-time algorithm. It would make sense, in the spirit of (Bernecker et al., 2009) for relational data, of first evaluating the frequency of one-item sets, and then use this information as a basis to discover itemsets of larger cardinality, making use of the probability scores for top-k retrieval. In other words, this chapter has described how to model probabilistic XML data, query it, and use this query capability as a general tool for various mining tasks. Further work could elaborate efficient algorithms for a specific mining task without necessary relying on the existing querying techniques.

Acknowledgments

This work would not have been possible without the previous joint work (Abiteboul et al., 2010) on aggregate queries with Serge Abiteboul, T.-H. Hubert Chan, and Werner Nutt. This work was supported by the European Research Council grant Webdam (under FP7), grant agreement 226513, and by the Dataring project of the French ANR. We are also grateful to the anonymous reviewers and the editor of this book, whose comments were of great help in improving the chapter.

References

Abiteboul, S., Chan, T.-H. H., Kharlamov, E., Nutt, W., & Senellart, P. (2010). Aggregate queries for discrete and continuous probabilistic XML. In *Proc. International Conference on Database Theory (ICDT)* (pp. 50–61). New York, NY: ACM.

Abiteboul, S., Hull, R., & Vianu, V. (1995). *Foundations of databases*. Reading, PA: Addison-Wesley.

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

Abiteboul, S., & Senellart, P. (2006). Querying and updating probabilistic information in XML. In *Proc. Extended Database Technology (EDBT)* (pp. 1059–1068). New York, NY: ACM.

Aggarwal, C. C. (Ed.). (2009). *Managing and mining uncertain data*. New York, NY: Springer.

Antova, L., Koch, C., & Olteanu, D. (2007). World-set decompositions: Expressiveness and efficient algorithms. In *Proc. International Conference on Database Theory (ICDT)* (pp. 194–208). New York, NY: ACM.

Ash, R. B., & Doléans-Dade, C. A. (2000). *Probability & measure theory*. San Diego, CA: Academic Press.

Barceló, P., Libkin, L., Poggi, A., & Sirangelo, C. (2009). XML with incomplete information: models, properties, and query answering. In *Proc. Symposium on Principles of Database Systems (PODS)* (pp. 237–246). New York, NY: ACM.

Benedikt, M., Kharlamov, E., Olteanu, D., & Senellart, P. (2010). Probabilistic XML via Markov chains. *Proceedings of the VLDB Endowment*, *3*(1).

Bernecker, T., Kriegel, H.-P., Renz, M., Verhein, F., & Züfle, A. (2009). Probabilistic frequent itemset mining in uncertain databases. In *Proc. Knowledge Discovery and Data Mining (KDD)* (p. 119-128). New York, NY: ACM.

Boulos, J., Dalvi, N. N., Mandhani, B., Mathur, S., Ré, C., & Suciu, D. (2005). MYSTIQ: a system for finding more answers by using probabilities. In *Proc. Special Interest Group on Management Of Data (SIGMOD)* (pp. 891–893). New York, NY: ACM.

Cheng, R., Kalashnikov, D. V., & Prabhakar, S. (2003). Evaluating probabilistic queries over imprecise data. In *Proc. Special Interest Group on Management Of Data (SIGMOD).* New York, NY: ACM.

Chomicki, J., & Libkin, L. (2000). Aggregate languages for constraint databases. In *Proc. Constraint Databases* (p. 131-154). New York, NY: Springer.

Codd, E. F. (1970). A relational model of data for large shared data banks. *Communications of the ACM*, *13*(6), 377–387.

Codd, E. F. (1975). Understanding relations (installment #7). *FDT - Bulletin of ACM SIGMOD*, *7*(3), 23–28. (Available from `http://portalparts.acm.org/990000/984403/fm/frontmatter.pdf`)

Codd, E. F. (1979). Extending the database relational model to capture more meaning. *ACM Transactions on Database Systems (TODS)*, *4*(4), 397–434.

Cohen, S., Kimelfeld, B., & Sagiv, Y. (2008). Incorporating constraints in probabilistic XML. In *Proc. Symposium on Principles of Database Systems (PODS)* (pp. 109–118). New York, NY: ACM.

Cohen, S., Kimelfeld, B., & Sagiv, Y. (2009). Running tree automata on probabilistic XML. In *Proc. Symposium on Principles of Database Systems (PODS)* (pp. 227–236). New York, NY: ACM.

Cohen, S., Nutt, W., & Sagiv, Y. (2006). Rewriting queries with arbitrary aggregation functions using views. *ACM Transactions on Database Systems (TODS)*, *31*(2), 672–715.

Date, C. J. (1981). *An introduction to database systems, 3rd edition*. Reading, PA: Addison-Wesley.

Dong, X. L., Halevy, A. Y., & Yu, C. (2009). Data integration with uncertainty. *VLDB Journal*, *18*(2), 469-500.

Fagin, R., Kolaitis, P. G., & Popa, L. (2005). Data exchange: getting to the core. *ACM Transactions on Database Systems (TODS)*, *30*(1), 174-210.

Fuhr, N., & Rölleke, T. (1997). A probabilistic relational algebra for the integration of information retrieval and database systems. *ACM Transactions on Information Systems (TOIS)*, *15*(1), 32–66.

Galindo, J., Urrutia, A., & Piattini, M. (2005). *Fuzzy databases: Modeling, design and implementation*. Hershey, PA: IGI Global.

Grant, J. (1977). Null values in a relational data base. *Information Processing Letters*, *6*(5), 156–157.

Green, T. J., & Tannen, V. (2006). Models for incomplete and probabilistic information. In *Proc. Extended Database Technology (EDBT) Workshops* (pp. 278–296). New York, NY: ACM.

Hollander, E., & Keulen, M. van. (2010). Storing and querying probabilistic XML using a probabilistic relational dbms. In *Proc. Management of Uncertain Data (MUD).*

Huang, J., Antova, L., Koch, C., & Olteanu, D. (2009). MayBMS: a probabilistic database management system. In *Proc. Special Interest Group on Management Of Data (SIGMOD)* (pp. 1071–1074). New York, NY: ACM.

Hung, E., Getoor, L., & Subrahmanian, V. S. (2003a). Probabilistic interval XML. In *Proc. International Conference on Database Theory (ICDT)* (pp. 358–374). New York, NY: ACM.

Hung, E., Getoor, L., & Subrahmanian, V. S. (2003b). PXML: A probabilistic semistructured data model and algebra. In *Proc. International Conference on Data Engineering (ICDE)* (p. 467). Washington, DC: IEEE.

Imieliński, T., & Lipski, W., Jr. (1984). Incomplete information in relational databases. *Journal of the ACM*, *31*(4), 761–791.

Imieliński, T., Naqvi, S. A., & Vadaparty, K. V. (1991). Querying design and planning databases. In *Proc. International Conference on Deductive and Object-Oriented Databases (DOOD)* (pp. 524–545). New York, NY: Springer.

ISO/IEC. (2008). *ISO/IEC 9075: SQL.* Geneva, Switzerland: International Standards Organization.

Keulen, M. van, Keijzer, A. de, & Alink, W. (2005). A probabilistic XML approach to data integration. In *Proc. International Conference on Data Engineering (ICDE)* (pp. 459–470). Washington, DC: IEEE.

Kharlamov, E., Nutt, W., & Senellart, P. (2010). Updating Probabilistic XML. In *Proc. Extended Database Technology (EDBT) Workshops.* New York, NY: ACM.

Kimelfeld, B., Kosharovski, Y., & Sagiv, Y. (2008). Query efficiency in probabilistic XML models. In *Proc. Special Interest Group on Management Of Data (SIGMOD)* (pp. 701–714). New York, NY: ACM.

Kimelfeld, B., Kosharovsky, Y., & Sagiv, Y. (2009). Query evaluation over probabilistic XML. *VLDB Journal*, *18*(5), 1117–1140.

Lafferty, J., McCallum, A., & Pereira, F. (2001). Conditional Random Fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. International Conference on Machine Learning (ICML).* San Fransisco, CA: Morgan Kaufmann.

Lakshmanan, L. V. S., Leone, N., Ross, R. B., & Subrahmanian, V. S. (1997). ProbView: A flexible probabilistic database system. *ACM Transactions on Database Systems (TODS)*, *22*(3), 419–469.

Li, T., Shao, Q., & Chen, Y. (2006). Pepx: a query-friendly probabilistic XML database. In *Proc. International Conference on Information and Knowledge Management (CIKM).*

Libkin, L., & Wong, L. (1996). Semantic representations and query languages for or-sets. *Journal of Computer and System Sciences*, *52*(1), 125–142.

Lopatenko, A., & Bertossi, L. E. (2007). Complexity of consistent query answering in databases under cardinality-based and incremental repair semantics. In *Proc. International Conference on Database Theory (icdt)* (p. 179-193). New York, NY: Springer.

Nayak, R. (2005). Knowledge discovery from XML documents. In *Encyclopaedia of data warehousing and data mining.* Hershey, PA: IGI Global.

Nierman, A., & Jagadish, H. V. (2002). ProTDB: Probabilistic data in XML. In *Proc. International Conference on Very Large Data Base (VLDB)* (pp. 646–657). San Fransisco, CA: Morgan Kaufmann.

Ré, C., Dalvi, N. N., & Suciu, D. (2006). Query evaluation on probabilistic databases. *IEEE Data Engineering Bulletin*, *29*(1), 25–31.

Ricci, F., Rokach, L., Shapira, B., & Kantor, P. (Eds.). (2010). *Recommender systems handbook*. New York, NY: Springer.

Sarma, A. D., Benjelloun, O., Halevy, A. Y., & Widom, J. (2006). Working models for uncertain data. In *Proc. International Conference on Data Engineering (ICDE)* (p. 7). Washington, DC: IEEE.

Sen, P., Deshpande, A., & Getoor, L. (2009). PrDB: managing and exploiting rich correlations in probabilistic databases. *VLDB Journal*, *18*(5), 1065–1090.

Senellart, P., & Abiteboul, S. (2007). On the complexity of managing probabilistic XML data. In *Proc. Symposium on Principles of Database Systems (PODS)* (pp. 283–292). New York, NY: ACM.

Senellart, P., & Souihli, A. (2010a, October). *ProApproX: A lightweight approximation query processor over probabilistic trees.* (Preprint)

Senellart, P., & Souihli, A. (2010b, October). Un système de gestion de données XML probabilistes. In *Bases de Données Avancées (BDA).* Toulouse, France. (Conference without formal proceedings. (Demonstration))

Tan, P.-N., Steinbach, M., & Kumar, V. (2005). *Introduction to data mining*. Reading, PA: Addison-Wesley.

Vardi, M. Y. (1982). The complexity of relational query languages (extended abstract). In *Proc. Symposium on Theory of Computing (STOC)* (pp. 137–146). New York, NY: ACM.

Widom, J. (2005). Trio: A system for integrated management of data, accuracy, and lineage. In *Proc. Conference on Innovative Data Systems Research (CIDR)* (pp. 262–276). Online Proceedings.

Zimányi, E. (1997). Query evaluation in probabilistic relational databases. *Theoretical Computer Science*, *171*(1–2), 179–219.

Additional Reading

Abiteboul, S., Chan, T.-H. H., Kharlamov, E., Nutt, W., & Senellart, P. (2010). Aggregate queries for discrete and continuous probabilistic XML. In *Proc. International Conference on Database Theory (ICDT)* (pp. 50–61). New York, NY: ACM.

Abiteboul, S., Hull, R., & Vianu, V. (1995). *Foundations of databases*. Reading, PA: Addison-Wesley.

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

Aggarwal, C. C. (Ed.). (2009). *Managing and mining uncertain data*. New York, NY: Springer.

Kimelfeld, B., Kosharovsky, Y., & Sagiv, Y. (2009). Query evaluation over probabilistic XML. *VLDB Journal*, *18*(5), 1117–1140.

Key Terms and Definitions

P-documents. A p-document is a tree with two types of nodes: ordinary and distributional. A p-document can be thought of as a probabilistic process that generates a random XML document in a conceptually simple way, namely, each distributional node chooses a subset of its children. Therefore, each distributional node of a p-document should specify the probability distribution of choosing a subset of its children in the above random process. There are several types of distributional nodes that differ from one another in how they specify probabilities.

Continuous P-documents. Continuous p-documents generalize p-documents to documents whose leaves are labeled with (representations of) probability distributions over the reals, instead of single values.

Tree-Pattern Queries. A tree-pattern query is a tree with two types of edges: child edges and descendant edges, and with a tuple of nodes defining its output. The nodes of the tree are labeled by a labeling function with either labels or variables, such that no variable occurs more than once, that is, join variables are not allowed. If the tuple of output nodes is empty tuple, the query is *Boolean*. A *single-path query* is a tree-pattern query whose set of edges is a linear order.

Aggregate Functions. An aggregate function maps a finite bag of values (e.g., rationals) into some domain (possibly the same or different). Standard aggregate functions are sum, count, min, countd (count distinct), and avg (average) under their usual semantics.

$\mathsf{FP}^{\#\mathsf{P}}$ Complexity Class. An $\mathbb{N}$-valued function f is in #P if there is a non-deterministic 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 $\mathsf{FP}^{\#\mathsf{P}}$ if it is computable in polynomial time using an oracle for some function in #P. A function is $\mathsf{FP}^{\#\mathsf{P}}$-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 $\mathsf{FP}^{\#\mathsf{P}}$ 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, hence #P-complete. Note also that #P-hardness clearly implies NP-hardness.

Data Complexity. It is usually assumed that databases are large while queries are typically very small. Therefore, a common approach to measure complexity of query evaluation, called *data complexity*, is to assume that the query is fixed and only the database is given as input.