Habilitation à diriger les recherches en informatique

XML probabiliste : Un modèle de données pour le Web

Probabilistic XML: A Data Model for the Web

Pierre Senellart

13 juin 2012

Jury

Serge Abiteboul	DR, INRIA Saclay ; Prof., Collège de France	
Bernd Amann	Prof., Université Pierre et Marie Curie	
Christoph Koch	Prof., EPFL	(rapporteur)
Gérard Memmi	Prof., Télécom ParisTech	
Frank Neven	Prof., Univ. Hasselt	(rapporteur)
Joachim Niehren	DR, INRIA Lille	(rapporteur)
Val Tannen	Prof., Univ. Pennsylvanie	(membre invité)

XML probabiliste : Un modèle de données pour le Web

Probabilistic XML: A Data Model for the Web

Pierre Senellart

Résumé

Les données extraites du Web sont chargées d'incertitude : elles peuvent contenir des contradictions ou résulter de processus par nature incertains comme l'intégration de données ou l'extraction automatique d'informations. Dans cette thèse d'habilitation, je présente les modèles de données XML probabilistes, la manière dont ils peuvent être utilisés pour représenter les données du Web, et la complexité de différentes opérations de gestion de données sur ces modèles. Je donne un état de l'art exhaustif du domaine, en insistant sur mes propres contributions. Je termine par un résumé de mes futurs projets de recherche.

Abstract

Data extracted from the Web often come with uncertainty: they may contain contradictions or result from inherently uncertain processes such as data integration or automatic information extraction. In this habilitation thesis, I present probabilistic XML data models, how they can be used to represent Web data, and the complexity of the different data management operations on these models. I give an exhaustive survey of the state-of-the-art in this field, insisting on my own contributions. I conclude with a summary of my research plans.

Mots clefs : données du Web, données probabilistes, World Wide Web, XML

Keywords: Web data, probabilistic data, World Wide Web, XML

Cette thèse d'habilitation est rédigée en anglais. Les deux premiers chapitres disposent de deux niveaux de lecture : (i) le cœur du texte, publié indépendamment (Kimelfeld et Senellart 2012), constitue une vue d'ensemble de l'état de l'art dans la gestion de données XML probabilistes, écrite conjointement avec Benny Kimelfeld ; (ii) je détaille aux endroits appropriés (par des encadrés « ✐ Contributions ») une partie de mes propres travaux de recherche, en relation avec les thèmes du texte principal.Le troisième chapitre présente un résumé de mes projets futurs de recherche.

This habilitation thesis is written in English. The first two chapters feature two layers of reading: (i) the core text, independently published (Kimelfeld and Senellart 2012), provides a survey of the state-of-the-art in probabilistic XML data management, jointly written with Benny Kimelfeld; (ii) I detail in relevant places (marked by highlighted "✐ Contributions" boxes) part of my own research works, in connection with the topics of the main text.The third chapter presents a summary of my research plans.

Cette thèse est rédigée à l'aide du système de composition de documents LaTeX, avec les polices de caractères Adobe Garamond Pro, Math Design, Myriad Pro et Bitstream Vera Sans Mono. Les bandes dessinées introduisant chaque chapitre sont extraites de *The Complete Peanuts*, de Charles M. Schultz, une collection publiée depuis 2004 par Fantagraphics Books. Ils sont reproduits ici en vertu du droit de citation.

This thesis is written with the help of the typesetting system LaTeX, in the font families Adobe Garamond Pro, Math Design, Myriad Pro, and Bitstream Vera Sans Mono. The comic strips that introduce each chapter are from *The Complete Peanuts*, by Charles M. Schultz, a series published since 2004 by Fantagraphics Books. They are reprinted here under fair use.

Contents

Acknowledgments

I have had the chance over the past ten years to work with a variety of individuals, not all of them were part of the works presented here. I am grateful to all of them, for what they have taught me. Let me still mention some of them, for their impact on both my research and my personal development. Serge Abiteboul, of course, was an ideal PhD adviser, and has remained a friend I immensely enjoy to carry out research with. I have learned much from Michael Benedikt over the past few years, on a variety of research projects we embarked on. I am impressed with his dedication and grateful for his trust. Benny Kimelfeld is the coauthor of the survey this thesis is based on. The PhD students I had the chance to work with (Alban Galland, Evgeny Kharlamov, Asma Souihli, Marilena Oita, Mouhamadou Lamine Ba, Yael Amsterdamer, Muhammad Faheem) also taught me about collaborating on research, and allowed me to participate in a broad diversity of topics.

I would also like to thank the members of my habilitation committee for agreeing to take on that charge, and, especially, Christoph Koch, Frank Neven, and Joachim Niehren, for reviewing this thesis.

A large part of my research activities has been carried out in the framework of the ERC Advanced Grant Webdam (Abiteboul, Senellart, and Vianu 2012), thanks Serge and the EU for making this possible!

I also acknowledge my colleagues at Télécom ParisTech in general, and in the DBWeb team in particular. I have very much enjoyed working in this environment.

Finally, my thoughts go to my friends and family. My sister Martine has been an anchor in my life throughout the years, and has contributed once again to this thesis her extensive knowledge of *Peanuts*-o-logy. My partner, Yann, in addition to being an important part of my life, has the particularity of often appearing in the Acknowledgments sections of my papers, for his keen insights on the topics we informally discuss. Thanks for bearing with me.

Chapter 1

An Uncertain World

Real-world data (say, from the World Wide Web) is often uncertain, because of the inherent uncertainty of the data itself, or of data collection, integration, and management processes. We explain in this chapter how probabilistic databases in general, and probabilistic XML in particular, can be used to formally manage this uncertainty.

1.1 From Uncertain Data to Probabilistic Databases

Data managed by modern database applications are often *uncertain*. A few examples are the following. When information from different sources is conflicting, inconsistent, or simply presented in incompatible forms, the result of integrating these sources necessarily involves uncertainty as to which fact is correct or which is the best mapping to a global schema. When data result from automatic and imprecise tasks, such as information extraction, data mining, or computer vision, it is commonly annotated by a score representing the *confidence* of the system in the correctness of the data. When data are gathered from sensor networks, they come with the inherent imprecision in the measurement of sensors. Even when data is generated by humans, they are not necessarily certain: diagnostics of diseases stored in a hospital database are affected by the imprecision of human judgment in addition to that of the diagnostics themselves. This ubiquity of uncertain data is all the truer when one deals with the World Wide Web, which is a heterogeneous collection of data that is constantly updated by individuals and automated processes.

Data uncertainty is often ignored, or modeled in a specific, per-application manner. This may be an unsatisfying solution in the long run, especially when the uncertainty needs to be retained throughout complex and potentially imprecise processing of the data. As an example, consider sensor data being gathered in a database, mined to extract interesting patterns, annotated by human experts, then integrated together with the result of other such analyses, independently made. Each of these steps, from the initial collection to the final integration, should be aware of the uncertain character of handled data; furthermore, each of these steps may even introduce further uncertainty. The goal of uncertain data management is to provide a unifying framework and a unifying system to handle the semantics of uncertainty, in the database itself. This goal is in line with the motivation behind DBMSs themselves, which were proposed in the 1970s as a uniform solution to the problem of managing data, while replacing previous systems that were tied to particular applications.

Naturally, there are various ways to model uncertainty. Examples include representation of missing information (from SQL NULLs to more elaborate models of *incomplete data* (Imieliński and Lipski 1984)), fuzzy logic and fuzzy sets (Galindo, Urrutia, and Piattini 2005), and the Dempster-Shafer theory (Zadeh 1986). In this dissertation, we consider *probabilistic* models that represent probability distributions over ordinary databases, and are based on the rich mathematical formalism of probability theory: *probabilities allow measuring the uncertainty*. Of course, quite a few real-life applications provide data that are probabilistic in nature. Examples include conditional random fields (Lafferty, McCallum, and Pereira 2001) (used in information extraction), statistics-based tasks involved in natural-language processing (Manning and Schutze 1999), or ranking mechanisms based on Markov chain theory and probabilistic fixpoint approaches, such as PageRank (Brin and Page 1998). But even when applications do not provide obvious probabilistic data, they often provide confidence scores that can be mapped to probability values.

Contributions

In a series of work on *graph mining*, I have used PageRank-like approaches to extract meaningful information from the Web graph or similar networks. In the following cases, the ranking scores obtained are or can be interpreted as probabilities, making the result of the algorithms a suitable dataset for probabilistic data management.

(Blondel, Gajardo, Heymans, Senellart, and Dooren 2004) We have generalized Kleinberg's HITS algorithm (Kleinberg 1999) to the comparison of nodes between two arbitrary graphs. In particular, we have applied the approach to synonym discovery in a dictionary graph (Senellart and Blondel 2008). The similarity scores can be interpreted as a probability distribution for a node in one of the graphs to be the best match of one in the second graph.

(Ollivier and Senellart 2007) We have introduced Green measures as a parameter-free tool for finding related nodes in a graph, with application to finding pages similar to a given page in Wikipedia. The Green measure centered at i can be thought of as the PageRank measure modified by feeding constant mass on node i at each iteration step.

(Vazirgiannis, Drosos, Senellart, and Vlachou 2008) We have shown how to predict the evolution of PageRank scores on the Web by using hidden Markov models.

(Galland, Abiteboul, Marian, and Senellart 2010) We have used a fixpoint computation approach to estimate the probability that a fact is true, when contradictory facts are stated by various sources, of various trustworthiness. Trust is also estimated in the process. This truth discovery technique has various applications, e.g., improving poll-based predictions by simultaneously identifying users of high quality.

(Suchanek, Abiteboul, and Senellart 2011) We have built a pseudo-probabilistic model, turning first-order formulas into probabilistic fixpoint computations, to match two ontology graphs. We thus obtain probabilities that one entity of the first ontology is equal to another one in the second, or that a relation of the first ontology is a subrelation of one in the second. Our approach is parameter-free, does not require any form of supervision, and works at large scale.

1.2 Representing and Querying Probabilistic Databases

A *probabilistic database* is, conceptually, a probability space over ordinary databases, each of which is called a *possible world* (Dalvi and Suciu 2007a). In practice, such a probability space is obtained by introducing uncertainty about the value (or existence) of individual data items. If there are many such uncertain data items, then the number of possible worlds may be too large to manage or even to store. However, applications usually make assumptions on the correlation among the uncertain items (e.g., independence), and such assumptions typically allow for a substantially smaller (e.g., logarithmic-size) representation of the probability space. Hence, from a database point of view, the goal is to provide a proper language and underlying data model to specify and represent probabilistic databases in a *compact* manner (e.g., by building in the model assumptions of independence).

But a database is not just about storage. A central role of a database is to provide a clean, easy, and general way of accessing its data (while abstracting away from the actual implementation). In particular, a database supports a high-level language like SQL or XQuery. In the context of probabilistic databases, the correspondent of querying is that of finding *events* and *inferring* their probabilities. Hence, we would like the database not just to "store probabilities," but to actually understand their semantics and support inference tasks. The common realization of that (Dalvi and Suciu 2007a; Kimelfeld, Kosharovsky, and Sagiv 2009; Nierman and Jagadish 2002; Hollander and Keulen 2010) is to allow the user to phrase ordinary queries (of the kind she would pose to an ordinary database), while the database associates each query answer with its computed probability (i.e., the probability that the answer holds true in a random possible world).

Finally, another central task of a database is to provide high performance for the operations it supports (e.g., query evaluation). This aspect is particularly challenging in the case of a probabilistic database, due to the magnitude of the actual probability space that such a database can (compactly) represent. As an example, for query evaluation (under the semantics mentioned in the previous paragraph), the baseline way of computing the probabilities is through the enumeration of all possible worlds, which is prohibitively intractable. Hence, we would like the operations to be performed on the compact representation itself rather than on the possible worlds. From the theoretical-complexity point of view, we require efficiency to be under the assumption that the input consists of the database in its compact form; in particular, "polynomial-time" is in the size of the compact representation, and not in that of the implied probability space. Not surprisingly, this requirement leads very quickly to computational hardness (Dalvi and Suciu 2007a) (and sometimes even hardness of approximation (Kimelfeld, Kosharovsky, and Sagiv 2009; Fagin, Kimelfeld, and Kolaitis 2010)). However, as we discuss throughout the thesis, in the case of probabilistic XML there are a few important settings where querying is tractable.

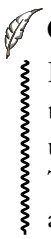
There is a rich literature on probabilistic relational databases (Widom 2005; Huang, Antova, Koch, and Olteanu 2009; Cheng, Singh, and Prabhakar 2005; Jampani, Xu, Wu, Perez, Jermaine, and Haas 2008; Sen, Deshpande, and Getoor 2009; Dalvi, Ré, and Suciu 2009; Suciu, Olteanu, Ré, and Koch 2011). In contrast, here we discuss *probabilistic XML* models, which represent probabilistic spaces over labeled trees. XML is naturally adapted to a number of applications where data is tree-like, including Web data or natural language parsing, and we give next specific examples of applications of probabilistic XML models. Later in this thesis (Section 2.5.3), we discuss the connection between probabilistic relational models and probabilistic XML models.

1.3 Probabilistic XML Applications

Following are concrete examples of applications or application areas where probabilistic XML is a natural data model and, moreover, the need to *query* probabilistic XML arises.

- **XML data integration.** Assume that a number of sources on the Web export XML information in potentially different schemas. To represent the result of the integration, we need a way to capture the uncertainty in the schema mappings, in deduplication, or in resolving conflicting information. This uncertainty can be characterized by probabilistic mappings (Fagin, Kimelfeld, and Kolaitis 2010) and probabilistic data integration rules (Keulen, Keijzer, and Alink 2005; Keulen and Keijzer 2009). The outcome of the integration process can naturally be viewed as probabilistic XML (which is useful to query, update, and so on).

- **Web information extraction.** Extracting information from Web data means detecting, in a Web page, instances of concepts, or relations between these instances, based on the content or structure of these Web pages. A typical output is therefore a tree-like document, with local annotations about extracted information. Current extraction techniques, whether they are unsupervised or rely on training examples, are by nature imprecise, and several possible annotations might be produced for the same part of the Web page, with confidence scores. This is for instance the case with conditional random fields for XML (Jousse, Gilleron, Tellier, and Tommasi 2006) that produce probabilistic labels for part of the original HTML document; probabilistic XML is a natural way to represent that.

  Contributions

  In (Senellart, Mittal, Muschick, Gilleron, and Tommasi 2008), we have applied conditional random fields for XML to discover the structure of deep Web response pages, using the output of a domain-specific gazetteer to guide the machine learning approach. The outcome is a wrapper assigning probabilities that part of a Web page is to be annotated with a domain concept.

- **Natural language parsing.** Parsing natural language consists in building syntax trees out of sentences. This is an uncertain operation, because of the complexity of the natural language, and its inherent ambiguity. Indeed, some sentences like "I saw her duck" have several possible syntax trees. A parser will typically rely on statistics gathered from corpora to assign probabilities to the different possible parse trees of a sentence (Manning and Schutze 1999). This probability space of parse trees can then be seen as probabilistic XML data (Cohen and Kimelfeld 2010).

- **Uncertainty in collaborative editing.** Consider users collaborating to edit documentation structured in sections, sections, paragraphs and so on, as in the online encyclopedia Wikipedia. In an open environment, some of these contributions may be incorrect, or even spam and vandalism. If we have some way to estimate the trustworthiness of a contributor, we can represent each individual edit as an uncertain operation on a probabilistic XML document that represents the integration of all previous edits (Abdessalem, Ba, and Senellart 2011).

Contributions

In (Abdessalem, Ba, and Senellart 2011), we have demonstrated a system that produces probabilistic XML documents representing the uncertain view of a Wikipedia page resulting of a series of uncertain edit operations. In (Ba, Abdessalem, and Senellart 2011), we have explained how this approach can be applied to the general setting of uncertain version control systems, simulating version control operations (addition, deletion, merge, etc.) by update operations on probabilistic XML data.

- **Probabilistic summaries of XML corpora.** Querying and mining a large corpus of XML documents (e.g., the content of the DBLP bibliography) can be time-consuming. If we are able to summarize this corpus as a compact probabilistic model (Abiteboul, Amsterdamer, Deutch, Milo, and Senellart 2012), namely probabilistic XML, we can then use this model to get (approximations of) the result of querying or mining operations on the original corpus.

Contributions

In (Abiteboul, Amsterdamer, Deutch, Milo, and Senellart 2012), we have shown how to assign optimal probabilities to tree automata describing the schema of an XML corpus, without or with integrity constraints. Optimality is defined in terms of likelihood of generation. As an application, we have demonstrated in (Abiteboul, Amsterdamer, Milo, and Senellart 2012) an intelligent auto-completion editor for XML, that takes into account a corpus of documents for suggesting the most likely completion of the current document.

We now move to a formal presentation of probabilistic XML and a description of the main results of the literature on managing probabilistic XML data.

Chapter 2

Probabilistic XML: Models and Complexity

This chapter surveys the state of the art in probabilistic XML data management. It is organized as follows. We first introduce basic concepts, mainly XML, probabilistic XML, p-documents, and ProTDB as our main example of a concrete p-document model (Section 2.1). Next, we talk about querying probabilistic documents in general, and within ProTDB in particular (Section 2.2). We then review and discuss additional models (Section 2.3) and additional problems of interest (Section 2.4). Finally, we discuss practical aspects of probabilistic XML systems (Section 2.5) and conclude (Section 2.6).

As a complement to this chapter, we maintain an updated list of resources (especially, a hyperlinked bibliography) pertaining to probabilistic XML, online at `http://www.probabilistic-xml.org/`.

2.1 Probabilistic XML

In this section, we describe the formal setting of this chapter, and in particular give the formal definitions of our basic concepts: an (ordinary) XML document, a probabilistic XML space, and the p-document representation of probabilistic XML.

2.1.1 XML Documents

We assume an infinite set Σ of *labels*, where a label in Σ can represent an XML tag, an XML attribute, a textual value embedded within an XML element, or the value of an attribute. The assumption that Σ is infinite is done for the sake of complexity analysis. An *XML document* (or just *document* for short) is a (finite) directed and ordered tree, where each node has a label from Σ. The label of a document node v is denoted by $\text{label}(v)$. We denote by $\mathbf{D}_\Sigma$ the (infinite) set of all documents.

As an example, the bottom part of Figure 2.1 shows a document d. In this figure, as well as in other figures, labels that represent textual values (e.g., "*car financing*") are written in italic font, as opposed to labels that represent tags (e.g., "title"), which are written in normal font. Note that the direction of edges is not explicitly specified, and is assumed to be downward. Similarly, order among siblings is assumed to be left-to-right.

2.1.2 px-Spaces

A *probabilistic XML space*, abbreviated *px-space*, is a probability space over documents. Although we will briefly discuss *continuous* px-spaces (Section 2.3.5), our focus is mainly on *discrete* px-spaces. So, unless stated otherwise, we will implicitly assume that a px-space is discrete. Specifically, we view a px-space as a pair $\mathscr{X} = (D, p)$, where D is a finite or countably infinite set of documents, and $p : D \to [0, 1]$ is a *probability function* satisfying $\sum_{d \in D} p(d) = 1$. The *support* of a px-space $\mathscr{X} = (D, p)$ is the set of documents $d \in D$, such that $p(d) > 0$. We say that the px-space $\mathscr{X}$ is *finite* if $\mathscr{X}$ has a finite support; otherwise, $\mathscr{X}$ is *infinite*.

When there is no risk of ambiguity, we may abuse our notation and identify a px-space $\mathscr{X}$ by the random variable that gets a document chosen according to the distribution of $\mathscr{X}$. So, for example, if $\mathscr{X} = (D, p)$ and d is a document, then $\Pr(\mathscr{X} = d)$ (in words, *the probability that $\mathscr{X}$ is equal to d*) is $p(d)$ if $d \in D$, and 0 otherwise.

2.1.3 p-Documents

A px-space is encoded by means of a *compact representation*. Later in this chapter, we will discuss the plethora of representation models proposed and studied in the literature. The basic notion underlying most of those models is that of a *p-document* (Kimelfeld and Sagiv 2007; Abiteboul, Kimelfeld, Sagiv, and Senellart 2009).

Contributions

This formal model was introduced in (Kimelfeld and Sagiv 2007). In (Abiteboul, Kimelfeld, Sagiv, and Senellart 2009), we have built on this to present a comprehensive view of existing probabilistic XML models, and of their respective expressiveness and compactness, leveraging results of (Kimelfeld and Sagiv 2007; Senellart and Abiteboul 2007).

Formally, a p-document is a tree $\mathscr{P}$ that is similar to an XML document, except that $\mathscr{P}$ has a distinguished set of *distributional nodes* in addition to the *ordinary nodes* (that have labels from Σ). The ordinary nodes of $\mathscr{P}$ may belong to documents in the encoded px-space. Distributional nodes, on the other hand, are used only for defining the probabilistic process that generates random documents (but they do not actually occur in those documents). As an example, Figure 2.1 shows a p-document $\mathscr{P}$, where the distributional nodes are the ones represented by boxes with rounded corners (and denoted by v_1, v_2, and so on). The words ind and mux inside those boxes will be discussed later. Each distributional node specifies a probability distribution over subsets of its children; later on, we will define several *types* of distributional nodes (like ind and mux), where each type defines the way these distributions are encoded. In the probabilistic process that generates a random document, a distributional node randomly chooses a subset of its children according to the distribution specified for that node. The root and leaves of a p-document are required to be ordinary nodes.

Next, we describe the px-space (D, p) defined by a p-document $\mathscr{P}$ by specifying a sampling process that generates a random document. Note that such a process well defines the px-space (D, p) as follows: D consists of all the documents that can be produced in this process, and $p(d)$ (where $d \in D$) is the probability that d is obtained in this process.

The random document is generated by the p-document $\mathscr{P}$ in two steps. First, each distributional node of $\mathscr{P}$ randomly chooses a subset of its children. Note that the choices of different nodes are not necessarily probabilistically independent. All the unchosen children and their descendants (even descendants that have been chosen by their own parents) are deleted. The second step removes all

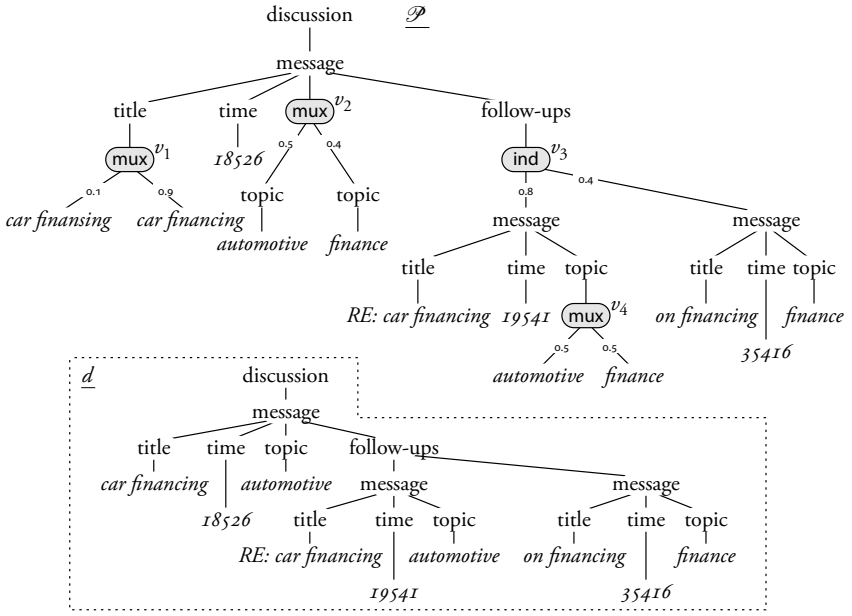


Figure 2.1: A p-document $\mathscr{P}$ in $\mathsf{PrXML}^{\{\mathsf{ind},\mathsf{mux}\}}$ (top) and a sample document d of $\mathscr{P}$ (bottom)

the distributional nodes. If an ordinary node u remains, but its parent is removed, then the new parent of u is the lowest ordinary node v of $\mathscr{P}$, such that v is a proper ancestor of u. Note that two different applications of the first step may result in the same random document generated (and for further discussion on that, see (Kimelfeld and Sagiv 2007)).

A Concrete Model: ProTDB

We now construct a concrete model of p-documents, namely, the ProTDB model (Nierman and Jagadish 2002). For that, we define two types of distributional nodes. Recall that when defining a type of distributional nodes, we need to specify the encoding and meaning of the random process in which a distributional node of that type selects children. In Section 2.3, we will define additional types of distributional nodes (hence, additional concrete models).

A ProTDB document has two types of distributional nodes.

- ind: A distributional node v of type ind specifies for each child w, the probability of choosing w. This choice is independent of the other choices of children, of either v or other distributional nodes in the p-document.
- mux: A distributional node v of type mux chooses at most one child w (that is, different children are mutually exclusive, hence the name mux) with a specified probability for w. We require the sum of probabilities along the children of v to be at most 1; the complement of this sum of probabilities is the probability that v chooses none of its children.

Example 1. *The top part of Figure 2.1 shows a ProTDB p-document $\mathscr{P}$. The type of each distributional node is written in the corresponding box. For instance, node v_1 is a distributional node of type* mux*; as shown by the numbers on its outgoing edges, v_1 chooses its left child and right child with probability* 0.1 *and* 0.9*, respectively. Note that the* mux *node v_2 chooses none of its children with probability* 0.1 *(*$= 1-0.4-0.5$*). Finally, observe that the* ind *node v_3 makes independent choices about its two children;*

for example, it chooses just *the left child with probability* $0.8 \times (1-0.4)$*, both children with probability* 0.8×0.4*, and no children at all with probability* $(1-0.8) \times (1-0.4)$.

In the bottom, Figure 2.1 shows a sample document d of $\mathscr{P}$. Let us now compute the probability of d. For d to be produced, the following independent events should take place:

- *v_1 chooses its right child. This event occurs with probability* 0.9.
- *v_2 chooses its left child. This event occurs with probability* 0.5.
- *v_3 chooses both of its children. This event occurs with probability* $0.8 \times 0.4 = 0.32$.
- *v_4 chooses its right child. This event occurs with probability* 0.5.

Hence, the probability of d is given by

$$\Pr(\mathscr{P} = d) = 0.9 \times 0.5 \times 0.32 \times 0.5 = 0.072\,.$$

We follow the conventional notation (Abiteboul, Kimelfeld, Sagiv, and Senellart 2009) that, given k types type_1, type_2, …, type_k of distributional nodes (such as ind, mux, and the types that we define later), $\mathsf{PrXML}^{\{\text{type}_1,\text{type}_2,\ldots,\text{type}_k\}}$ denotes the model of p-documents that use distributional nodes only among type_1, type_2, …, type_k. Hence, under this notation ProTDB is the model $\mathsf{PrXML}^{\{\mathsf{ind},\mathsf{mux}\}}$ (and for the p-document $\mathscr{P}$ of Figure 2.1 we have $\mathscr{P} \in \mathsf{PrXML}^{\{\mathsf{ind},\mathsf{mux}\}}$). Observe that $\mathsf{PrXML}^{\{\mathsf{ind},\mathsf{mux}\}}$ strictly contains $\mathsf{PrXML}^{\{\mathsf{ind}\}}$, $\mathsf{PrXML}^{\{\mathsf{mux}\}}$ and $\mathsf{PrXML}^{\{\}}$ (which is the set $\mathbf{D}_\Sigma$ of ordinary documents).

2.2 Query Evaluation

In this section, we discuss a central aspect in the management of probabilistic XML—query evaluation. In general, a query Q maps a document d to a value $Q(d)$ in some domain dom_Q; that is, a query is a function $Q : \mathbf{D}_\Sigma \to dom_Q$. As an example, in the case of a *Boolean* query, dom_Q is the set $\{\mathbf{true}, \mathbf{false}\}$; in that case we may write $d \models Q$ instead of $Q(d) = \mathbf{true}$ (and $d \not\models Q$ instead of $Q(d) = \mathbf{false}$). In the case of an *aggregate* query, dom_Q is usually the set $\mathbb{Q}$ of rational numbers. Later on, we discuss additional types of queries.

A px-space $\mathscr{X}$ and a query Q naturally define a probability distribution over dom_Q, where the probability of a value $a \in dom_Q$ is given by $\Pr(Q(\mathscr{X}) = a)$. We usually follow the conventional semantics (Dalvi and Suciu 2007a) that, when evaluating Q over $\mathscr{X}$, the output represents that distribution. For example, if Q is a Boolean query, then the goal is to compute the number $\Pr(\mathscr{X} \models Q)$.

2.2.1 Query Languages

We now describe the languages of queries that capture the focus of this chapter: tree-pattern queries, monadic second-order queries, and aggregate queries.

Tree-Pattern Queries

Tree-pattern queries (a.k.a. *twig queries* (Bruno, Koudas, and Srivastava 2002; Amer-Yahia, Cho, Lakshmanan, and Srivastava 2001)), or just *tree patterns* for short, correspond to the navigational fragment of XPath restricted to child and descendant edges. Specifically, a tree pattern is a Boolean query that is represented by a tree t with *child* and *descendant* edges. In our figures, child and descendant edges are depicted by single and double lines, respectively. Each node of the tree t is

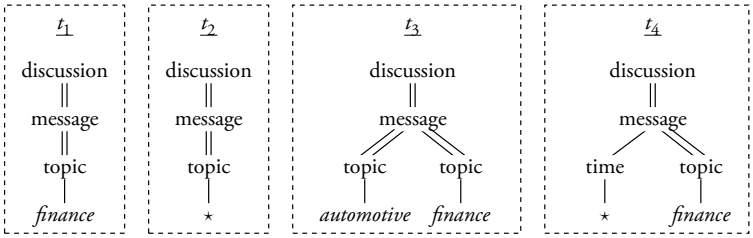


Figure 2.2: Tree patterns

labeled with either a label of Σ or with the special *wildcard* symbol $\star$ (and we assume that $\star \notin \Sigma$). A *match* of a tree pattern t in a document d is a mapping μ from the nodes of t to those of d, such that μ maps root to root, child edges to edges, and descendant edges to paths (with at least one edge); furthermore, μ preserves labels of Σ, that is, for a node v of t, if $\text{label}(v) \neq \star$ then $\text{label}(v) = \text{label}(\mu(v))$. Note that a tree pattern ignores the order among siblings in a document. (Queries that take sibling order into account will be discussed in the next section.)

Example 2. *Four tree patterns, $t_1, \ldots, t_4$, are shown in Figure 2.2. Child and descendant edges are represented by single and double lines, respectively. As in documents (and p-documents), edges are implicitly directed top down. As specific examples, let us consider the patterns t_1 and t_4. The pattern t_1 says that some message in the document has a topic descendant (where this topic can be that of the message or of a descendant message) with a child finance. The pattern t_4 is the same, except that it also requires the message to have a time child, and the time child to have a child (any child, as indicated by $\star$) of its own.*

Tree patterns are often used not just as Boolean queries, but also as queries that produce tuples of nodes (or tuples of labels). Informally speaking, these tuples are obtained by projecting the matches to a selected sequence of nodes of the tree pattern. For the sake of simplicity, here we restrict the discussion to the Boolean case. It is important to note, though, that under the standard notion of query evaluation for such queries[1] (Dalvi and Suciu 2007a), evaluating a non-Boolean tree pattern reduces in polynomial time to evaluating a Boolean one (Kimelfeld, Kosharovsky, and Sagiv 2009).

Monadic Second-Order Tree Logic (MSO)

A language that is far more expressive than tree patterns is that of *Monadic Second-Order tree logic* (MSO). A query in MSO is a Boolean formula over the document nodes. The vocabulary includes two binary relations over nodes x and y: "x is the parent of y," denoted $E(x,y)$, and "x is a following sibling of y," denoted $y < x$. For each label $\lambda \in \Sigma$, the vocabulary includes also the unary relation "λ is the label of x," denoted $\lambda(x)$. The formula is in first-order logic, extended with quantification over set variables. This quantification allows, among other things, to express conditions on the set of all ancestors or descendants of a node, or on that of all nodes following a given node in document order. For a more formal definition the reader is referred to the vast literature on MSO for trees (e.g., Neven and Schwentick (Neven and Schwentick 2002)).

Example 3. *For illustration, the following MSO query says that there is a message with two descendants*

[1] Under this notion, the output consists of every possible result tuple $\vec{a}$ and its marginal probability $\Pr(\vec{a} \in Q(\mathcal{X}))$.

that are consecutive sibling messages on the topic finance.

$$\exists x, y_1, y_2 [\mathit{message}(x) \wedge \mathit{message}(y_1) \wedge \mathit{message}(y_2) \\ \wedge \underline{\mathit{descendant}}(x, y_1) \wedge \underline{\mathit{descendant}}(x, y_2) \wedge \underline{\mathit{next\text{-}sibling}}(y_1, y_2) \\ \wedge \underline{\mathit{finance\text{-}topic}}(y_1) \wedge \underline{\mathit{finance\text{-}topic}}(y_2)]$$

In the formula above, $\underline{\mathit{descendant}}(x, y)$ *is phrased in MSO as follows.*

$$\forall S[S(x) \wedge \forall z_1, z_2(S(z_1) \wedge E(z_1, z_2) \rightarrow S(z_2)) \rightarrow S(y)]$$

Similarly, $\underline{\mathit{next\text{-}sibling}}(y_1, y_2)$ *is given by* $y_1 < y_2 \wedge \neg \exists z[y_1 < z < y_2]$*. Finally,* $\underline{\mathit{finance\text{-}topic}}(y)$ *is phrased in MSO as follows.*

$$\exists z, w[E(y, z) \wedge E(z, w) \wedge \mathit{topic}(z) \wedge \mathit{finance}(w)]$$

MSO queries are closely related to the notion of a *(bottom-up) nondeterministic tree automaton* (NTA). Specifically, every MSO query can be translated into an NTA, such that the documents that satisfy the MSO query are precisely those that are accepted by the NTA; conversely, every NTA can be similarly translated into an MSO query (Neven and Schwentick 2002; Doner 1970; Thatcher and Wright 1968).

Join Queries

Both tree patterns and MSO queries can be extended by adding *value joins* that test whether two nodes (for tree patterns), or two first-order variables (for MSO), have the same label. Value joins are fairly commonly used in XPath[2]; for instance, they allow us to dereference identifiers used as foreign keys.

Example 4. *The following query in MSO extended with the same-label predicate tests whether two messages that are descendant of each other have the same topic:*

$$\exists x_1, x_2, x_3, y_1, y_2, y_3 [\mathit{message}(x_1) \wedge \mathit{message}(y_1) \wedge \underline{\mathit{descendant}}(x_1, y_1) \\ \wedge E(x_1, x_2) \wedge \mathit{topic}(x_2) \wedge E(x_2, x_3) \\ \wedge E(y_1, y_2) \wedge \mathit{topic}(y_2) \wedge E(y_2, y_3) \\ \wedge \mathit{same\text{-}label}(x_3, y_3)]$$

Aggregate Queries

In this chapter, an *aggregate function* is a function α that takes as input a set V of document nodes, and returns as output a numerical (rational) number $\alpha(V) \in \mathbb{Q}$. Some of the aggregate functions we consider, like sum, need to assume that the label of a node is a number; to accommodate that, we fix a function num over the document nodes, such that $\mathrm{num}(v) = \mathrm{label}(v)$ if $\mathrm{label}(v)$ is a number, and otherwise, we arbitrarily determine $\mathrm{num}(v) = 0$. Specifically, we will discuss the following aggregate functions.

- Count: $\mathsf{count}(V) \stackrel{\text{def}}{=} |V|$.
- Count distinct: $\mathsf{countd}(V) \stackrel{\text{def}}{=} |\{\mathrm{label}(v) \mid v \in V\}|$; that is, $\mathsf{countd}(V)$ is the number of distinct labels that occur in V, regardless of the multiplicity of these labels.

[2] The first version of the XPath language only supports a limited form of value joins, but this restriction is lifted in the latest version.

- Sum: $\mathsf{sum}(V) \stackrel{\text{def}}{=} \sum_{v \in V} \mathrm{num}(v)$.
- Average: $\mathsf{avg}(V) \stackrel{\text{def}}{=} \mathsf{sum}(V)/|V|$; if V is empty, then $\mathsf{avg}(V)$ is *undefined*.
- Min/max: $\mathsf{min}(V) \stackrel{\text{def}}{=} \min_{v \in V} \mathrm{num}(v)$, $\mathsf{max}(V) \stackrel{\text{def}}{=} \max_{v \in V} \mathrm{num}(v)$.

An *aggregate query* applies an aggregate function to the set of nodes that is selected by another query (of a different type). Specifically, here we consider aggregate queries that we write as $\alpha \circ t[w]$, where α is an aggregate function, t is a tree pattern, and w is a node of t. The evaluation of $\alpha \circ t[w]$ over a document d results in the number $\alpha(V)$, where V is the set of nodes v of d, such that there exists a match μ of t in d with $\mu(w) = v$; that is:

$$\alpha \circ t[w](d) \stackrel{\text{def}}{=} \alpha(\{v \mid \mu(w) = v \text{ for some match } \mu \text{ of } t \text{ in } d\})$$

Example 5. *Consider the tree pattern t_2 of Figure 2.2, and let w be the wildcard (denoted $\star$) node. When applied to the document d of Figure 2.1, the query* $\mathsf{count} \circ t_2[w]$ *returns* 3*, which is the number of nodes with a "topic" parent. In contrast,* $\mathsf{countd} \circ t_2[w](d)$ *is* 2*, which is the number of* distinct *topics (i.e., distinct labels of nodes with a "topic" parent) in d.*

As another example, consider the tree pattern t_4 of Figure 2.2 and, again, let w be the wildcard node. The query $\mathsf{min} \circ t_4[w]$ *returns the earliest time of a message that has a descendant message on the topic finance; hence,* $\mathsf{min} \circ t_4[w](d) =$ *"18526".*

2.2.2 Complexity for ProTDB

Nierman and Jagadish (2002) studied the evaluation of (non-Boolean) tree patterns without projection, and showed computability in polynomial time. Although projection leads to hardness in the relational probabilistic model (Dalvi and Suciu 2007a), Kimelfeld, Kosharovsky, and Sagiv (2009) showed that tree patterns with projection, and in particular Boolean tree patterns, can be evaluated in polynomial time in ProTDB (Kimelfeld, Kosharovsky, and Sagiv 2009). Cohen, Kimelfeld, and Sagiv (2009b) extended this result to MSO queries. The main reason behind this tractability is that it is possible to evaluate queries directly over a ProTDB tree in a bottom-up manner, making use of the locality of both the p-document and the query. This can be done using dynamic programming for tree patterns (Kimelfeld and Sagiv 2007), and through the computation of a product automaton of the query and the p-document in the MSO case (Benedikt, Kharlamov, Olteanu, and Senellart 2010).

Contributions

We observe in (Benedikt, Kharlamov, Olteanu, and Senellart 2010) that ProTDB p-documents, and extensions thereof, are simple cases of probabilistic tree automata, that can be combined with the tree automaton of a query to compute the probability of this query.

Theorem 6 (Cohen, Kimelfeld, and Sagiv 2009b). *Let Q be an MSO query (e.g., a tree pattern). The problem "compute* $\Pr(\mathscr{P} \models Q)$ *given* $\mathscr{P} \in \mathsf{PrXML}^{\{\mathsf{ind},\mathsf{mux}\}}$*" is in polynomial time.*

Observe that Theorem 6 is phrased in terms of *data complexity* (Vardi 1982), which means that the query is held fixed. As mentioned in (Kimelfeld, Kosharovsky, and Sagiv 2009), the evaluation of tree patterns becomes intractable if the query is given as part of the input. Actually, it was shown (Cohen, Kimelfeld, and Sagiv 2009b; Kimelfeld, Kosharovsky, and Sagiv 2009) that over ProTDB the evaluation of tree patterns, and even MSO queries, is *fixed-parameter tractable* (abbr. FPT) (Downey

and Fellows 1999), which means that only the coefficients (rather than the degree) of the polynomial depend on the query[3] (hence, FPT is stronger than "polynomial data complexity"). Nevertheless, while for tree patterns this dependence is "merely" exponential, for general MSO queries this dependence is not any elementary function (unless P = NP), since that is already the case when the p-document is ordinary (deterministic) (Meyer 1975; Frick and Grohe 2004).

Tractability (in terms of data complexity) is lost when tree patterns are extended with (value) joins (Abiteboul, Chan, Kharlamov, Nutt, and Senellart 2010). This is not surprising, for the following reason. Tree patterns with joins over trees can simulate Conjunctive Queries (CQs) over relations. Moreover, tree patterns with joins over $\mathsf{PrXML}^{\{\mathsf{ind}\}}$ can simulate CQs over "tuple-independent" probabilistic relations (Dalvi and Suciu 2007a). But the evaluation of CQs over tuple-independent probabilistic databases can be intractable even for very simple (and small) CQs (Dalvi and Suciu 2007a). Interestingly, it has been shown that adding *any* (single) join to *any* tree pattern results in a query that is intractable, unless that query is equivalent to a join-free pattern (Kharlamov, Nutt, and Senellart 2011).

Contributions

In our works on probabilistic XML, we have payed special attention to join queries, that are often used in practice. The hardness of queries with joins is a simple observation (Abiteboul, Chan, Kharlamov, Nutt, and Senellart 2010), as already noted, but the interesting part of the following theorem is the dichotomy, obtained in (Kharlamov, Nutt, and Senellart 2011), of a much simpler nature than in the relational case (Dalvi and Suciu 2007b). However, the dichotomy result is only known for queries with single joins at the moment, extension to queries with arbitrarily many joins is open.

Theorem 7 (Kharlamov, Nutt, and Senellart 2011). *If Q is a tree pattern with a single join predicate, then one of the following holds.*

1. *Q is equivalent to a tree pattern (hence, can be evaluated in polynomial time).*
2. *The problem "compute $\Pr(\mathscr{P} \models Q)$ given $\mathscr{P} \in \mathsf{PrXML}^{\{\mathsf{ind},\mathsf{mux}\}}$" is #P-hard.*

Recall that #P is the class of functions that count the number of accepting paths of the input of an NP machine (Valiant 1979); this class is highly intractable, since using an oracle to a #P-hard function one can solve in polynomial time every problem in the polynomial hierarchy (Toda and Ogiwara 1992).

Next, we discuss aggregate queries. Cohen, Kimelfeld, and Sagiv (2009a) showed that for the aggregate functions count, min, and max, the evaluation of the corresponding aggregate queries is in polynomial time for $\mathsf{PrXML}^{\{\mathsf{ind},\mathsf{mux}\}}$. (That result of Cohen, Kimelfeld, and Sagiv (2009a) is actually for a significantly broader class of queries, which they refer to as "constraints.") Note that, for these specific functions, the number of possible results (numbers) q is polynomial in the size of the input p-documents; hence the evaluation of an aggregate query Q reduces (in polynomial time) to the evaluation of $\Pr(Q(\mathscr{P}) = q)$.

Theorem 8 (Cohen, Kimelfeld, and Sagiv 2009a). *Let Q be the aggregate query $\alpha \circ t[w]$. If α is either* count, min *or* max, *then the problem "compute $\Pr(Q(\mathscr{P}) = q)$ given $\mathscr{P} \in \mathsf{PrXML}^{\{\mathsf{ind},\mathsf{mux}\}}$ and $q \in \mathbb{Q}$" is in polynomial time.*

[3] For a formal definition of FPT the reader is referred to Flum and Grohe's book (Flum and Grohe 2006).

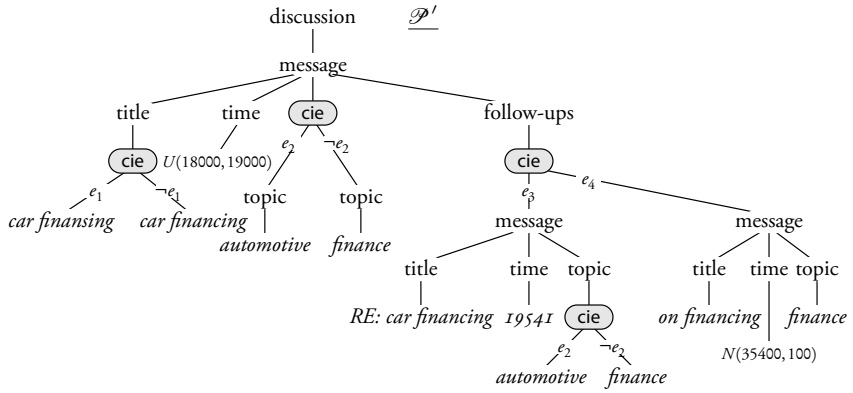


Figure 2.3: A continuous p-document $\mathscr{P}'$ in $\mathsf{PrXML}^{\{\mathsf{cie}\}}$ with $\Pr(e_1) = 0.1$, $\Pr(e_2) = 0.5$, $\Pr(e_3) = 0.8$, $\Pr(e_4) = 0.4$

Note that an immediate consequence of Theorem 8 is that we can evaluate, in polynomial time, Boolean queries like $\mathsf{count} \circ t[w] > q$ (i.e., where equality is replaced with a different comparison operator). Unfortunately, this result does not extend to the aggregate functions countd, sum and avg.

Contributions

We have obtained in (Abiteboul, Chan, Kharlamov, Nutt, and Senellart 2010) a comprehensive characterization of the complexity of aggregate queries over ProTDB (and more complex models), building on the work of (Cohen, Kimelfeld, and Sagiv 2009a).

Theorem 9 (Abiteboul, Chan, Kharlamov, Nutt, and Senellart 2010; Cohen, Kimelfeld, and Sagiv 2009a). *For each α among* countd, sum *and* avg *there is an aggregate query $Q = \alpha \circ t[w]$, such that the problem "determine whether $\Pr(Q(\mathscr{P}) = q) > 0$ given $\mathscr{P} \in \mathsf{PrXML}^{\{\mathsf{ind},\mathsf{mux}\}}$ and $q \in \mathbb{Q}$" is NP-complete.*

A particularly interesting fact that is shown by Theorems 8 and 9 is that there is an inherent difference between the complexity of count and countd when it comes to query evaluation over $\mathsf{PrXML}^{\{\mathsf{ind},\mathsf{mux}\}}$.

2.3 Additional p-Documents and Extensions

We now discuss additional representation systems for probabilistic XML. Some of these systems are p-document models with additional kinds of distributional nodes, and other systems are extensions of the p-document concept. We discuss the expressive power of these representation systems, and the complexity of query answering.

2.3.1 Long-distance dependencies

The mux and ind distributional nodes encode *local* dependencies between nodes, in the sense that the presence of a node in the document depends just on the presence of its parent and (in the case of mux) its siblings. However, it is often desired to represent *long-distance* dependencies to capture correlations among nodes of arbitrary locations in the document tree. Towards that, we introduce

new kinds of distributional nodes. Assume a finite set $\{e_1 \dots e_n\}$ of independent Boolean random variables (called *Boolean events*), and a probability $\Pr(e_i)$ for each of these e_i. We define two new kinds of distributional nodes:

- cie (Abiteboul and Senellart 2006; Abiteboul, Kimelfeld, Sagiv, and Senellart 2009): A distributional node v of type cie specifies for each child w of v a *conjunction of independent events* e_k or their negation $\neg e_k$ (e.g., $e_2 \wedge \neg e_5 \wedge e_6$).

  Contributions

  (Abiteboul and Senellart 2006) was our first work on probabilistic XML. We compared this model to a simple one, based on ind nodes, highlighting the expressiveness and compactness of the former. This was inspired by the use of c-tables (Imieliński and Lipski 1984) for representing incomplete data, probabilistic versions of which were independently proposed by Green and Tannen (2006).

- fie (Kharlamov, Nutt, and Senellart 2010): A distributional node v of type fie specifies for each child w of v an arbitrary propositional formula on the e_is (e.g., $e_2 \vee (e_3 \wedge \neg e_7)$).

  Contributions

  We have introduced this model in (Kharlamov, Nutt, and Senellart 2010) to obtain a fully tractable model for updates, at the cost of query efficiency. The full power of propositional formulas as annotations of nodes is needed for some applications, such as uncertain version control (Ba, Abdessalem, and Senellart 2011).

Recall from Section 2.1 that, to define the semantics of a type of distributional node, we need to specify how a random subset of children is chosen by a node of that type. For cie and fie, the specification is as follows. At the beginning of the process, we draw a random truth assignment τ to the events $e_1, \dots, e_n$, independently of one another and according to the probabilities $\Pr(e_1), \dots, \Pr(e_n)$. Then, each distributional node selects the children that are annotated by a formula that evaluates to true under τ. (We then proceed to the second step, as described in Section 2.1.)

Example 10. *An example p-document $\mathcal{P}'$ of* $\mathsf{PrXML}^{\{\mathsf{cie}\}}$ *is shown in Figure 2.3. Disregard for now the leaf nodes under "time" nodes (these nodes contain continuous distributions that will be discussed in Section 2.3.5). The p-document $\mathcal{P}'$ is somewhat similar to $\mathcal{P}$ of Figure 2.1: there is uncertainty in the title of the first message, in its topic, and in the existence of the two follow-up messages, which are independent of each other. However, there is also a fundamental difference. The topic of the first follow-up is correlated with that of the original message: either both are set to "*automotive*" or both are set to "*finance*." This reflects what a topic extraction system might do, if it has a global view of the whole discussion.*

We now look at the relative expressiveness and succinctness of p-documents defined with ind, mux, cie, and fie distributional nodes. In terms of expressiveness, $\mathsf{PrXML}^{\{\mathsf{ind},\mathsf{mux}\}}$, $\mathsf{PrXML}^{\{\mathsf{cie}\}}$, and $\mathsf{PrXML}^{\{\mathsf{fie}\}}$ are all able to represent all finite probability distributions over documents and are therefore equivalent (Abiteboul, Kimelfeld, Sagiv, and Senellart 2009) (as already noted, this is not the case for $\mathsf{PrXML}^{\{\mathsf{ind}\}}$, $\mathsf{PrXML}^{\{\mathsf{mux}\}}$ and, obviously, $\mathsf{PrXML}^{\{\}}$). However, in terms of succinctness the picture is different: while there is a polynomial-time transformation of a $\mathsf{PrXML}^{\{\mathsf{ind},\mathsf{mux}\}}$ p-document into an equivalent $\mathsf{PrXML}^{\{\mathsf{cie}\}}$ p-document, the converse is not true (Kimelfeld, Kosharovsky, and Sagiv 2008).

Similarly, PrXML$^{\{cie\}}$ is a subset of PrXML$^{\{fie\}}$, but a transformation from PrXML$^{\{fie\}}$ into PrXML$^{\{cie\}}$ entails an inevitable exponential blowup (Kharlamov, Nutt, and Senellart 2010).

The families PrXML$^{\{cie\}}$ and (a fortiori) PrXML$^{\{fie\}}$ are thus exponentially more succinct than ProTDB. However, this succinctness comes at a cost: query evaluation is now intractable. More precisely, *every* (Boolean) tree-pattern query is *#P*-hard over PrXML$^{\{cie\}}$ (and PrXML$^{\{fie\}}$), except for some trivial cases (Kimelfeld, Kosharovsky, and Sagiv 2008; Kimelfeld, Kosharovsky, and Sagiv 2009). The situation in PrXML$^{\{cie\}}$ is essentially the same as that in PrXML$^{\{fie\}}$, although a few specific types of queries are tractable over PrXML$^{\{cie\}}$ and yet intractable over PrXML$^{\{fie\}}$: projection-free tree patterns with joins (Senellart and Abiteboul 2007), and expected values for some types of aggregate queries (Abiteboul, Chan, Kharlamov, Nutt, and Senellart 2010, 2011).

Contributions

Intractability of PrXML$^{\{fie\}}$ is a simple consequence (Kharlamov, Nutt, and Senellart 2010) of the results of (Toda and Ogiwara 1992). Tractable subcases for PrXML$^{\{cie\}}$ are of special interest. In (Senellart and Abiteboul 2007), we have showed tractability of an even larger query languages, *locally monotone* queries, when projections are forbidden. In other terms, this just means that collecting the *provenance* (Green, Karvounarakis, and Tannen 2007) of a query is easy, evaluating it is hard. Expected values of simple aggregate queries are tractable (Abiteboul, Chan, Kharlamov, Nutt, and Senellart 2010, 2011) because two sum operators (or one sum and one integral) commute.

The intractability of querying p-documents with long-distance dependencies discussed above concerns the computation of the *exact* probability of a query. It makes sense to look also at approximation algorithms (Kimelfeld, Kosharovsky, and Sagiv 2009). The simplest way to approximate query probability is by Monte-Carlo sampling: pick a random document, evaluate the query, and iterate. The approximated probability will then be the ratio of draws for which the probability evaluated to true. This approach yields a polynomial-time algorithm for obtaining an *additive* approximation of the query probability; that is, a number that is guaranteed, with high confidence, to be in the interval $[p-\varepsilon; p+\varepsilon]$ around the exact probability p. Using other means (Karp, Luby, and Madras 1989), in the case of tree patterns over PrXML$^{\{cie\}}$ it is also possible to obtain a (polynomial-time) *multiplicative* approximation (i.e., a number in the interval $[(1-\varepsilon)p, (1+\varepsilon)p]$) (Kimelfeld, Kosharovsky, and Sagiv 2009).

2.3.2 Conditional models

As mentioned earlier, a central drawback in the ProTDB model (i.e., PrXML$^{\{ind,mux\}}$) and some other models proposed in the literature (e.g., (Hung, Getoor, and Subrahmanian 2003)) is the assumption of probabilistic independence among probabilistic choices; in turn, this assumption is the key reason for the tractability of query evaluation (Kimelfeld and Sagiv 2007). However, even simple additional information about the database may give rise to intricate correlations. As a simple example, consider again the p-document in Figure 1. Even if we do not know the exact structure of the messages (hence, we use probabilistic rather than deterministic XML), it is likely that we know the total number of messages, and precisely (with no uncertainty involved). This new detail introduces dependency among the children of v_3, since now a random world cannot have too many (or too few) messages altogether. A more intricate statement can be the fact that at least 90% of the messages with the topic *automotive* have one or more *automotive* follow-ups; note that this statement implies correlation between the distributional nodes v_2 and v_4.

To incorporate such additional information, Cohen, Kimelfeld, and Sagiv (2009a) suggested

to specify *constraints* in addition to the p-document. They presented a language for specifying constrains that may involve aggregate functions (e.g., "the total number of messages is 392," and "at least 80% of the messages have follow-ups").[4] Formally, a *Probabilistic XML Database* (PXDB) is a pair $(\mathscr{P}, \mathscr{C})$, where $\mathscr{P}$ is a p-document and $\mathscr{C}$ is a set of *constraints*. The px-space that is defined by a PXDB $(\mathscr{P}, \mathscr{C})$ is the sub-space of $\mathscr{P}$ conditioned on the satisfaction of each constraint of $\mathscr{C}$ (in other words, we restrict the px-space to the possible worlds that satisfy $\mathscr{C}$, and normalize the probabilities). Cohen et al. gave polynomial-time algorithms for various central tasks, such as *sampling* and *querying*, where their queries are tree patterns with some aggregate functions (that include count, and min/max).[5] Similar tractability results have been shown for the case where both constraints and queries are phrased in MSO (Cohen, Kimelfeld, and Sagiv 2009b).

2.3.3 Recursive Markov chains

In principle, p-documents provide means of representing arbitrary *finite* px-spaces. Some applications, however, require the ability to represent infinite sets of possible worlds. Consider again the example document of Figure 2.1; all such documents describing email discussions conform to the following schema, given as a DTD:

```
discussion: (message*)
   message: (title, time, topic?, follow-ups?)
follow-ups: (message*)
```

There are infinitely many documents conforming to this DTD, of arbitrarily large depth and width. In order to represent a discussion in which the number of messages and the structure of the discussion itself is fully uncertain, we need to be able to model, in a concise manner, infinite px-spaces.

The formalism of *recursive Markov chains* (Etessami and Yannakakis 2009) is used for describing recursive probabilistic processes. Alternatives are described using a Markov chain, where each node in the chain can be a call to another (or the same) chain. This formalism naturally lends itself to the representation of potentially infinite px-spaces, as shown by Benedikt, Kharlamov, Olteanu, and Senellart (2010). That work studies the tractability of MSO queries over px-spaces represented by recursive Markov chains (and restrictions thereof). In particular, recursive Markov chains that are *hierarchical* (i.e., when there are no cycles in the call graph) are tractable if we assume that all arithmetic operations have unit cost.[6] Hierarchical Markov chains can be seen as a generalization of p-documents defined with directed acyclic graphs instead of trees, a model introduced in (Cohen, Kimelfeld, and Sagiv 2009b; Benedikt, Kharlamov, Olteanu, and Senellart 2010). If we further restrict recursive Markov chains so that no Markov chain is called at two different positions (they are thus *tree-like*), we obtain a fully tractable model that generalizes $\mathsf{PrXML}^{\{\mathsf{mux,ind}\}}$ (and even more succinct models, e.g., $\mathsf{PrXML}^{\{\mathsf{exp}\}}$ (Kimelfeld, Kosharovsky, and Sagiv 2008)).

[4] For the precise specification of this language, see (Cohen, Kimelfeld, and Sagiv 2009a).

[5] This language allows for nesting of queries, and the reader is referred to (Cohen, Kimelfeld, and Sagiv 2009a) for the exact details.

[6] Without this assumption, we lose tractability because the exact probability of a query may require exponentially many bits in the size of the representation.

Contributions

In (Benedikt, Kharlamov, Olteanu, and Senellart 2010) we have shown how to describe infinite sets of possible worlds, and explained which tractability results carried over to these infinite models. Tree automata techniques, as well as the results of (Etessami and Yannakakis 2009), were used to show tractability of MSO over subclasses of recursive Markov chains.

2.3.4 SCFGs

A *Context-Free Grammar* (CFG) specifies a process of producing parse trees for strings in a nondeterministic manner; indeed, a specific string may have multiple (even infinitely many) parse trees, since multiple production rules can be specified for a nonterminal. A *stochastic* (or *probabilistic*) *Context-Free Grammar* (SCFG) is similar to a CFG, except that the rules are augmented with probabilities; that is, the production of a nonterminal becomes a probabilistic, rather than a nondeterministic, process.

When given a string, an SCFG implies a probability space over the possible parse trees of the string (where the probability of a parse tree corresponds to the confidence of the SCFG in that tree). Since this space comprises of labeled trees, we can view it as a (possibly infinite) px-space, on which we can evaluate XML queries (e.g., "find each noun phrase that forms an object for the verb *likes*"). Cohen and Kimelfeld (Cohen and Kimelfeld 2010) studied the problem of evaluating a tree-pattern query over the px-space that is represented by an SCFG and a string. In particular, they showed that this task is tractable for the class of *weakly linear SCFGs* (that generalizes popular normal forms like linear SCFGs, and Chomsky or Greibach normal forms). It follows from known results (Etessami and Yannakakis 2009) that, in the general case, query probabilities do not have a polynomial-size bit representation, and can even be irrational.

2.3.5 Continuous distributions

So far, all probabilistic XML models we have considered represent *discrete* probability distributions, where the uncertainty is either in the structure of the document or in the choice of a value from a finite collection of options. But some sources of uncertainty, such as the imprecision in sensor measurements, are essentially *continuous*. So, following (Abiteboul, Chan, Kharlamov, Nutt, and Senellart 2010, 2011) we introduce the possibility of labeling leaves of p-documents with not only constant values, but *continuous probability distributions* of values (as usual, represented in some compact manner). For example, we might say that a given leaf represents a uniform distribution between two constants.

Example 11. *Consider again the p-document $\mathcal{P}'$ of Figure 2.3. Two of the "time" nodes have for leaf a continuous distribution. The first one, $U(18000, 19000)$ represents a uniform distribution in the interval $[18000; 19000]$, which is adapted to the case when nothing else is known about the timestamp, perhaps because of a coarse granularity in the way the message metadata was displayed. The second distribution, $N(35400, 100)$ is a Gaussian centered around 35400 and with a standard deviation of 100. Such a timestamp might arise from a known imprecision in the date of the computer system that produced the timestamp. One can check that the document d of Figure 2.1 is one of the possible worlds represented by $\mathcal{P}'$ (but of course, it has a zero probability due to the continuous distributions).*

Observe that we cannot use our current formalism of a px-space to define the semantics of a p-document with continuous values, since our px-space is discrete, and in particular, is defined by

means of a probability of each possible world. Nevertheless, px-spaces can be properly extended to a continuous version by constructing a σ-algebra of sets of possible worlds, and define a probability measure over this σ-algebra, as done by Abiteboul, Chan, Kharlamov, Nutt, and Senellart (2011). When this is done, we can investigate the complexity of query evaluation, as usual. Tree patterns are not of much interest in this case, because if a query node is matched against a node with continuous distribution, the probability of this match is usually zero. But of course, aggregate queries make sense. As shown by Abiteboul, Chan, Kharlamov, Nutt, and Senellart (2011), the tractability of aggregate queries with functions such as count, min, or max extends from (discrete) ProTDB to the continuous case, as long as the class of probability distributions present in the p-document can be efficiently convoluted, summed, integrated, and multiplied. This is for instance the case of distributions defined by piecewise polynomials, a generalization of uniform distributions.

Contributions

One of the motivations of (Abiteboul, Chan, Kharlamov, Nutt, and Senellart 2010, 2011) was to provide a formal semantics for continuous probabilistic (tree) models and for querying them. The fact that they essentially do not introduce any complexity outside of the need of (symbolically or numerically) evaluating basic operations on continuous distributions, basically means that all existing query evaluation techniques can be used. Tractability critically relies on the independence of these distributions; it is not trivial to extend the model to capture dependent distributions, e.g., those obtained by updating the tree using, basic update operations that rely on locator queries (see next).

2.4 Other Problems of Interest

In the previous sections, we discussed the task of query evaluation over different models of probabilistic XML. Here, we discuss additional tasks. Specifically, we address *updating* and *typing*, which are classical XML operations. We also discuss *compression* (the problem of finding a representation of a smaller size), and *top-k querying* (retrieving the most probable answers to a tree-pattern or a keyword-search query). Finally, we briefly list additional tasks that are mostly left as future research.

2.4.1 Updates

In update languages like *XUpdate* or the *XQuery Update Facility*, the specification of update operations entails *locator queries* that indicate, as XPath or XQuery expressions, the locations where data are to be inserted, modified, or deleted. An elementary probabilistic update operation can thus be defined as consisting of a locator query, a specification of the operation to be performed at matched locations (e.g., a tree to be inserted), and a probability that the update should be performed (provided that the locator query matches). Such an operation has been studied by Abiteboul, Kimelfeld, Sagiv, and Senellart (2009). The semantics of updates is defined as for queries: the result of an update on a probabilistic database should be a representation of a probabilistic space obtained from the original probabilistic space by applying the update on every possible world. Again, we want to avoid the exponential enumeration of possible worlds and perform the update directly on the original probabilistic document. Updates are of particular interest since they can be seen as a fundamental mechanism for constructing a probabilistic XML document: a sequence of uncertain update operations applied to a deterministic XML document (Abdessalem, Ba, and Senellart 2011).

Limiting our study to ProTDB and models with long-distance dependencies, we observe the following tradeoff on update tractability (Kharlamov, Nutt, and Senellart 2010), in terms of data

complexity:

- The result of an update operation is computable in polynomial time over ProTDB for a restricted set of non-branching tree pattern queries (specifically, those without descendant edges or those whose locator query returns the node at the bottom of the tree pattern).
- In general, computing the result of an update operation over ProTDB is intractable.
- The result of an update operation is computable in polynomial time over the family $\mathsf{PrXML}^{\{\mathsf{fie}\}}$, for updates defined by tree-pattern queries with joins.

The reason for the tractability of updates in $\mathsf{PrXML}^{\{\mathsf{fie}\}}$ (while querying operations are hard) is that updates do not entail computation of probabilities; we just manipulate event formulas without computing their probabilities.

Updating probabilistic XML documents highlights the following issue in models different from ProTDB and $\mathsf{PrXML}^{\{\mathsf{fie}\}}$: the model may lack the property of being a *strong representation system* (Imieliński and Lipski 1984) for the query language used in locator queries; this means that it is impossible to represent the output of a query (or the result of an update based on this query language) in the model. This is the case for ProTDB extended with continuous value distributions, and the language of aggregate tree-pattern queries (or even tree-pattern queries with inequalities). To be able to apply updates on such probabilistic models, the challenge is to define generalizations of these models (and of the corresponding querying techniques) that are strong representation systems.

Contributions

As already noted, we have carried out extensive research on updating probabilistic XML documents (Senellart and Abiteboul 2007; Abiteboul, Kimelfeld, Sagiv, and Senellart 2009; Kharlamov, Nutt, and Senellart 2010) and on applications thereof (Abdessalem, Ba, and Senellart 2011; Ba, Abdessalem, and Senellart 2011). This aspect, fundamental for many applications (data warehousing, evolving p-documents, uncertain updates applied to deterministic databases, etc.), has been neglected in many other works on probabilistic databases. The best-studied probabilistic relational models, for instance, tuple-independent and block-independent-disjoint databases (Dalvi and Suciu 2007a), are not strong representation systems for simple conjunctive queries, due to their inability to represent complex correlations. In contrast, ProTDB, a very simple probabilistic XML model, is a strong representation system for arbitrary locally monotone queries (Abiteboul, Kimelfeld, Sagiv, and Senellart 2009).

2.4.2 Typing

Typing an XML document, that is, testing whether the document is valid against a schema defined in some schema language (e.g., DTD), is another fundamental data-management problem in XML. Similarly to Boolean querying, typing a probabilistic XML document should return a probability, namely, the probability that a random document is valid. As shown by Cohen, Kimelfeld, and Sagiv (2009b), when the schema can be defined by a deterministic bottom-up tree automaton (which is the case for DTDs, disregarding for now keys and foreign keys), computing the probability that a ProTDB p-document is valid is in polynomial time in the size of both the p-document and the schema. Essentially, this computation is done by running the automaton over the p-document, maintaining on the way some data structures that allow us to compute the probability that a node

has type q given the corresponding probabilities of its children. This result can be generalized in a number of ways. First, tractability extends to computing the probability of a fixed query (say, a tree pattern) in the probabilistic space that is restricted to only those worlds that are valid against a schema (Cohen, Kimelfeld, and Sagiv 2009b). Second, the data model can be generalized to recursive Markov chains, and we basically have tractability in the same classes of recursive Markov chains where MSO query answering is tractable (Benedikt, Kharlamov, Olteanu, and Senellart 2010). Third, adding constraints (such as keys and foreign keys) renders typing intractable, though it is still tractable to test whether the probability of being valid against a schema with constraints is exactly one (Cohen, Kimelfeld, and Sagiv 2009b).

2.4.3 Compression

A fundamental advantage of using probabilistic XML models, such as ProTDB, is their potential compactness in representing probabilistic spaces. Depending on the application, obtaining such a compact model might not be straightforward. The direct translation of a set of possible worlds with probabilities into a $\mathsf{PrXML}^{\{\mathsf{mux},\mathsf{ind}\}}$ document, for instance, simply enumerates all possible worlds as children of a mux node and has the same size as the original space. The *compression* or *simplification* problem (Keulen, Keijzer, and Alink 2005) is to obtain, given a probabilistic XML document, another more compact document that defines the same px-space.

In ProTDB, a basic operation that can be used to simplify a p-document is to push distributional nodes down the tree whenever possible, merging ordinary nodes in the process (Veldman, Keijzer, and Keulen 2009). Another direction is to apply regular XML compression techniques (Buneman, Grohe, and Koch 2003) to compress the probabilistic tree into a probabilistic DAG while retaining querying tractability (assuming unit-cost arithmetics), as discussed in Section 2.3.3. Veldman, Keijzer, and Keulen (2009) explored the combination of probabilistic XML simplification techniques with ordinary XML compression, demonstrating gain in the size of the representation.

2.4.4 Top-k Queries

Chang, Yu, and Qin (2009) studied the problem of finding, in a probabilistic XML document, the *top-k* query answers, that is, the k answers with the highest probabilities (where k is a specified natural number). Their model of probabilistic XML is ProTDB, and as queries they considered projection-free path patterns. Another type of a top-k query arises in *keyword search*. Information retrieval by keyword search on probabilistic XML has been studied by Li, Liu, Zhou, and Wang (2011). Specifically, they perform keyword search in the ProTDB model by adopting the notion of *Smallest Lower Common Ancestor* (SLCA) (Xu and Papakonstantinou 2005), which defines when an XML node constitutes an answer for a keyword-search query. More particularly, the problem they explore is that of finding the k nodes with the highest probabilities of being SLCAs in a random world.

2.4.5 Open Problems

We now discuss important open problems around management operations on probabilistic XML. Despite the existence of techniques for compressing ProTDB documents (Veldman, Keijzer, and Keulen 2009), we lack a good understanding on when compression is possible and whether it is possible to obtain an *optimal* representation (with respect to compactness) of a px-space, in ProTDB and other models. A fundamental problem related to this one concerns *equivalence* of probabilistic XML documents: decide whether two representations define the same px-space (Keulen, Keijzer,

and Alink 2005). As shown in (Senellart 2007), this problem admits a randomized polynomial-time decision procedure for PrXML$^{\{cie\}}$ when p-documents are shallow. This gives some hope of obtaining, in practice, a more systematic procedure for minimizing the size of a p-document. Nevertheless, the exact complexities of the equivalence problem, of testing optimality, and of minimization itself, remain open problems.

Contributions

Probabilistic XML was motivated in my PhD thesis (Senellart 2007) by data warehousing applied to understanding the deep Web. Equivalence of p-documents is of critical importance to reduce the size of the probabilistic database, that would otherwise grow (sometimes unnecessarily) exponentially with the number of updates.

Compressing a discrete px-space into a compact p-document is somewhat akin to the problem of XML schema inference from XML data (Bex, Neven, and Vansummeren 2007): in both cases, the goal is to obtain a compact model of a set of documents. There are two differences, however. First, an XML schema represents a set of XML documents, while a p-document represents a probabilistic distribution thereof. Second, it is assumed that XML schema inference generalizes the observation of the example documents and that some documents valid against the schema are not present in the original collection, while compression preserves the px-space. Relaxing this last assumption leads to the problem of *probabilistic schema inference*, that is, learning a probabilistic model, with potential generalization, for a corpus of XML documents. A first work in this direction is by Abiteboul et al. (Abiteboul, Amsterdamer, Deutch, Milo, and Senellart 2012), where the skeleton of the schema is given, and probabilities are learned to optimize the likelihood of the corpus. Adapting XML schema inference techniques to directly generate probabilistic models would allow us to generalize any collection of XML documents as a probabilistic XML document.

The focus of most of the literature on probabilistic XML is on modeling and querying, while only little exploration has been done on other aspects of probabilistic XML management. One of the important aspects that deserve further exploration is that of *mining*, namely, discovering important patterns and trends (e.g., frequent items, correlations, summaries of data values, etc.) in probabilistic XML documents. Kharlamov and Senellart (Kharlamov and Senellart 2011) discuss how some mining tasks can be answered using techniques of probabilistic XML querying. Nevertheless, it is to be explored whether other techniques (e.g., based on ordinary frequent itemset discovery) can provide more effective mining.

Contributions

The algorithms we provide in (Kharlamov and Senellart 2011) for mining tasks, though quite naïve, show that some mining problems over probabilistic XML models (say, ProTDB) are at least tractable.

2.5 Practical Aspects

In this section, we discuss some practical aspects of probabilistic XML management. We first consider system architecture and indexing, and then elaborate on practical challenges that remain to be addressed towards developing a full-fledged database-management system for probabilistic XML. To the best of our knowledge, up to now only prototypical systems have been developed.

2.5.1 System Architecture

The first question is that of the general architecture of a probabilistic XML system: should it be *(a)* built on top of a probabilistic relational database system, *(b)* based on a query-evaluation engine for ordinary XML, *or (c)* engineered from scratch to easily accommodate the existing algorithmic approaches for probabilistic XML? We overview these three approaches, pointing to preliminary work, and noting advantages and shortcomings of each.

Over a Probabilistic Relational Engine

Much effort has been put on building efficient systems for managing probabilistic relational data. These systems include Trio (Widom 2005), MayBMS (Huang, Antova, Koch, and Olteanu 2009) and its query evaluator SPROUT (Olteanu, Huang, and Koch 2009). In turn, these systems are usually built on top of an ordinary relational database engine. Leveraging these efforts to the probabilistic XML case makes sense, and basically amounts to encoding probabilistic XML data into probabilistic tables, and tree-pattern queries into conjunctive queries. This direction is explored by Hollander and van Keulen (Hollander and Keulen 2010) with Trio, where feasibility is demonstrated for different kinds of XML-to-relation encodings. However, the relational queries that result from those encodings are of a specific form (e.g., inequalities are used to encode descendant queries). It turns out that typical relational optimizations are not always available for the resulting queries.

On top of an XML Query Engine

Alternatively, it is possible to rely on native XML database systems (such as eXist[7] or MonetDB/XQuery[8]) to evaluate queries over probabilistic XML documents, delegating components such as indexing of document structure and query optimization to the underlying XML database engine. It requires either modifying the internals of the XML query evaluation engine to deal with probabilities, or being able to rewrite queries over probabilistic XML documents as queries over ordinary documents. The latter approach is demonstrated by Senellart and Souihli (2011); there, tree-pattern queries with joins over p-documents of $\mathsf{PrXML}^{\{\mathsf{cie}\}}$ are rewritten into XQuery queries that retrieve each query match, along with a propositional formula that represents the probability of the match. All XML processing is therefore handed out to the XQuery query engine, and the problem is reduced to probability evaluation of propositional formulas.

Contributions

(Senellart and Souihli 2011) demonstrated the performance of approximation algorithms for answering queries over probabilistic XML documents. In an extension still in progress, we show that relying on an external XML query engine allows to obtain excellent efficiency in probabilistic query evaluation and to focus on the hard part of the problem, evaluating the probability of the lineage of a query.

Independent Implementation

The previous two architectures do not make use of the specificities of probabilistic XML, and in particular, of the techniques that have been developed for querying probabilistic XML. An alternative is thus to design a probabilistic XML system around one or more of these techniques

[7] http://exist.sourceforge.net/

[8] http://monetdb.project.cwi.nl/monetdb/XQuery/

(e.g., bottom-up dynamic programming (Kimelfeld and Sagiv 2007)), and thereby utilize the known algorithms at query time (Kimelfeld, Kosharovsky, and Sagiv 2008). The downside of this approach is that these techniques are main-memory intensive. Furthermore, the implemented system is typically applicable to only a limited probabilistic model[9], and to a limited class of queries[10].

2.5.2 Indexing

We now consider *indexing* as a mean of enhancing the efficiency of query evaluation over probabilistic XML. When a probabilistic XML system is implemented on top of an XML database system, we can rely on this system to properly index the tree structure and content. Still remains the issue of providing efficient access to the probabilistic annotations.

The PEPX system (Li, Shao, and Chen 2006) proposes to index ProTDB documents in the following manner: instead of storing with each child of a mux or ind node the probability of being selected by its parent, store the marginal probability that the child exists. Coupled with indexing of the tree structure, it allows much more efficient processing of simple queries, since a single access suffices to retrieve the probability of a node, and accessing all ancestors of this node is not required. This approach has also been taken by Li, Wang, Xin, Zhang, and Qiu (2009) who adapted the TwigStack algorithm (Bruno, Koudas, and Srivastava 2002) to the evaluation of projection-free patterns in a ProTDB document.

We believe probabilistic XML indexing can be leveraged beyond this, though. An interesting direction would be to combine structure-based indexing with probability-based indexing. Such an approach has the potential of enhancing the efficiency of finding the most probable answers (Chang, Yu, and Qin 2009) or answers with a probability above a specified threshold (Kimelfeld and Sagiv 2007).

2.5.3 Remaining Challenges

We now highlight some of the challenges that remain towards implementing a full-fledged system for managing probabilistic XML.

We first discuss the choice of method for query evaluation. Depending on the data model in use, and depending on the query language, we have a variety of techniques, exact or approximate: bottom-up algorithm in the absence of long-distance correlations (Kimelfeld and Sagiv 2007), naïve enumeration of all possible worlds, Monte-Carlo sampling, relative approximation (Kimelfeld, Kosharovsky, and Sagiv 2008), and so on. Each of these has specific particularities in terms of the range of query and data it can be applied to, its evaluation cost, and its approximation guarantee. Hence, it is likely that some methods are suitable in some cases and other methods are suitable in others. A good system should have a wealth of evaluation techniques and algorithms, and should be able to make proper decisions on which technique to use for providing a quick and accurate result. For example, the system may be given precision boundaries, and it should then select the most efficient approximation that guarantees these boundaries. Alternatively, given a time budget, a system should be able to select an exact or approximation technique (as precise as possible) for performing query evaluation within that budget. This process can be carried out at the level of the

[9] (Kimelfeld, Kosharovsky, and Sagiv 2008) supports just ProTDB documents, though it should be possible to use a similar bottom-up approach for hierarchical Markov chains (Benedikt, Kharlamov, Olteanu, and Senellart 2010) and to support continuous distributions (Abiteboul, Chan, Kharlamov, Nutt, and Senellart 2011).

[10] (Kimelfeld, Kosharovsky, and Sagiv 2008) supports just tree patterns, but it should also be possible to extend it to MSO by combining the algorithm of (Cohen, Kimelfeld, and Sagiv 2009b) and a toolkit such as Mona (Henriksen, Jensen, Jrgensen, Klarlund, Paige, Rauhe, and Sandholm 1995) for converting queries into tree automata.

whole query, or at the level of each sub-query. For instance, in some cases, it may be beneficial to combine probabilities that are computed (for different parts of the query and/or the document) by deploying different techniques. This suggests relying on cost-based, optimizer-like, query planning where each implementation of a (sub-)query evaluation is associated with an estimated cost, of both time and approximation. First steps are highlighted in (Souihli 2011).

Contributions

We are currently conducting extensive experiments (not yet published). These experiments show that such an optimizer is indeed capable of choosing an appropriate evaluation strategy for a given lineage formula, and that the optimal strategy (enumeration of possible worlds, Monte Carlo sampling, fancy algorithm for approximating DNF formulas, inclusion–exclusion principle, etc.) varies dramatically from query to query.

There is also a need for a deeper understanding of the connection between probabilistic XML and probabilistic relational data. This is obviously critical if one is to implement a probabilistic XML system on top of a probabilistic relational database. It is important in other architectures as well, for identifying techniques in the relational setting that carry over to the XML setting. This is not so straightforward. It is of course easy to encode trees into relations or relations into trees, but in both cases the encoding has special shapes: relations encoding trees are tree-like (with treewidth (Robertson and Seymour 1984) one) and relations encoded as trees are shallow and have repetitive structure. Typical query languages are different, too: tree-pattern queries or MSO on one side, conjunctive queries or the relational algebra on the other. When trees are encoded into relations, tree-pattern queries become a particular kind of conjunctive queries, involving hierarchically structured self joins, a class for which it is not always possible to obtain efficient query plans over arbitrary databases (Suciu, Olteanu, Ré, and Koch 2011). Some results from the probabilistic XML setting (such as the bottom-up evaluation algorithm for ProTDB) have no clear counterpart in the relational world, and vice versa. A unifying view of both models would help towards building systems for managing both probabilistic relational and XML data.

The last challenge we highlight is that of optimizing query evaluation by reusing computed answers of previous queries. This can be seen as a case of *query answering using views*, a problem that has been extensively studied in the deterministic XML setting (Xu and Özsoyoglu 2005; Cautis, Deutsch, and Onose 2008; Afrati, Chirkova, Gergatsoulis, Kimelfeld, Pavlaki, and Sagiv 2009). There is little known on whether and how (materialized) views can be used for query answering in the probabilistic XML setting, though Cautis and Kharlamov (2011) give a preliminary study of the problem in the setting of ProTDB, showing that the major challenge is not retrieving query answers, but computing their probabilities.

2.6 Conclusion

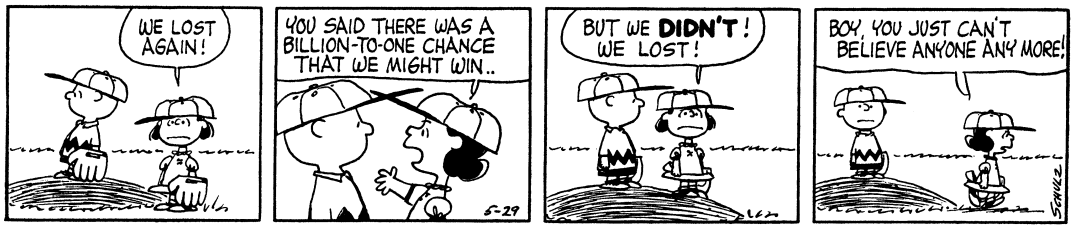
We reviewed the literature on probabilistic XML models, that are essentially representation systems for compactly encoding probability distributions over labeled trees. A variety of such representation systems have been proposed, and each provides a different trade-off between expressiveness and compactness on the one hand, and management complexity on the other hand. Specifically, ProTDB (Nierman and Jagadish 2002) and some of its extensions (e.g., ProTDB augmented with constraints or continuous distributions, and tree-like Markov chains) feature polynomial-time querying for a rich query language (MSO, or aggregate queries defined by tree-patterns). In contrast, query evaluation is intractable in other models such as $\mathsf{PrXML}^{\{\mathsf{fie}\}}$ (that allows for correlation among

arbitrary sets of nodes) or arbitrary recursive Markov chains (that can represent spaces of unbounded tree height or tree width).

We mentioned various open problems throughout this chapter. Two of these deserve particular emphasis. First, the connection to probabilistic relational models needs better understanding, from both the theoretical viewpoint (e.g., what makes tree-pattern queries over ProTDB tractable, when they are encoded into relations?) and the practical viewpoint (e.g., can we build on a system such as Trio (Widom 2005) or MayBMS (Huang, Antova, Koch, and Olteanu 2009) to effectively manage probabilistic XML data?). Second, further effort should be made to realize and demonstrate the ideal of using probabilistic XML databases, or probabilistic databases in general, to answer data needs of applications (rather than devising per-application solutions). We discussed some of the wide range of candidate applications in the introduction. We believe that the research of recent years, which is highly driven by the notable popularity of probabilistic databases in the database-research community, constitutes a significant progress towards this ideal, by significantly improving our understanding of probabilistic (XML) databases, by developing a plethora of algorithmic techniques, and by building prototype implementations thereof.

Chapter 3

Reconciling Web Data Models with the Actual Web



My existing research so far has focused on two aspects of Web data management (Abiteboul, Manolescu, Rousset, Rigaux, and Senellart 2012): models of uncertain Web data (with probabilistic XML) and methods for extracting meaningful content from the World Wide Web. These two research topics have been mostly disconnected. The main item on my research agenda is to reconcile them, and to show that existing formal models for Web data (probabilistic databases in particular) are natural and effective tools for representing and managing the results of Web mining tasks. We distinguish two general areas: concrete applications of probabilistic database techniques to Web data management problems, and improving the collection of Web content using formal models.

3.1 Probabilistic Databases: A Tool for Web Data

Probabilistic Databases vs. Ad-Hoc Management of Uncertainty for Data Warehousing. The main motivation for using probabilistic databases is to rely on a proper tool for managing uncertainty in complex processes, without having to select at each step the most likely result, but keeping all candidates and formally estimating their likelihood throughout the process. The effectiveness of probabilistic databases at this task has surprisingly never been demonstrated, however. We plan to formally assess the difference in quality that can be obtained in data warehousing tasks, comparing a probabilistic database approach to ad-hoc management of uncertainty. This is in the spirit of (Detwiler, Gatterbauer, Louie, Suciu, and Tarczy-Hornoch 2009) that looked at the effectiveness of formal uncertainty management techniques in ranking query results in scientific databases. The authors of this work have shown that such methods are indeed useful for practical applications (in this case, discovering functions of proteins), though probabilistic rankings do not necessarily outperform more ad-hoc scoring techniques.

Deriving Probabilistic XML Documents from Existing XML Corpora. XML corpora are concrete collections of XML documents. Probabilistic XML is a model of probability distributions over XML documents. By inferring the best probabilistic model of an XML corpus, we can apply probabilistic XML querying techniques directly on this model for a variety of statistics gathering, data visualization, and software testing applications.

Contributions

The case when the structure of the model is known, and probabilities are missing, has been dealt with in (Abiteboul, Amsterdamer, Deutch, Milo, and Senellart 2012; Abiteboul, Amsterdamer, Milo, and Senellart 2012).

Managing Uncertainty in Version Control using Probabilistic Database Techniques. Wikipedia can be seen as a version control system, in which each revision is uncertain and depends on the trustworthiness of the contributor. This uncertainty can naturally be represented as probabilistic, and the tree-like structure of articles lends itself to modeling with probabilistic XML. Version control operations (merging, updates, conflict resolution) can now be translated into operations on probabilistic XML documents.

Contributions

Preliminary steps are in (Abdessalem, Ba, and Senellart 2011; Ba, Abdessalem, and Senellart 2011).

Probabilistic Models for the Wisdom of the Crowd Crowd data sourcing (Deutch and Milo 2012) relies on the critical assumption that, although individuals are reliable, and the information they provide may be uncertain, the crowd as a whole is reliable. Probabilistic models are ideal for modeling this uncertainty: information about a given data item that individuals were polled about can for instance be represented as a normal distribution, whose mean and standard deviation are derived from the answers to the poll. Problems of determining the trust in some information, or of which question to ask to which individual to raise our confidence in it, can then be formalized as probabilistic data management questions.

3.2 Formal Models to Improve Web Content Collection

Database Theory and Static Analysis to the Rescue of Deep Web Content Acquisition. Accessing the deep Web, Web databases hidden behind Web forms, is a daunting task. Forms have to be semantically analyzed, result pages have to be wrapped, and queries have to be rewritten using only relevant Web sources. We plan to tackle this practical problem using tools from database theory and static analysis. For instance, the literature on query answering under access limitations can serve to rewrite and optimize queries over deep Web sources, and analysis of JavaScript validation code attached to Web pages can help inferring constraints on form schemas.

Contributions

On the database theory side, we have shown in (Senellart and Gottlob 2008; Gottlob and Senellart 2010) how to determine the optimal translation between two different representations of the same information, such as two different deep Web sources. Practical applications of this framework are still open, though. We have characterized in (Benedikt, Gottlob, and Senellart 2011) the complexity of determining whether a Web form is relevant to a particular query. Applications to query optimization over the deep Web are work in progress.

On the static analysis side, we have demonstrated in (Benedikt, Furche, Savvides, and Senellart 2012) a basic system that extracts simple constraints outside of JavaScript code. Though much remains to be done, results are very promising (100% precision in the identified constraints; reasonable recall of around 60%).

Application-Aware Web Crawling. Current-day Web crawlers are unaware of the kind of Web applications they are currently crawling: they process in the same way wikis, blogs, news sites, or statically edited content. This is not optimal, however: on a wiki, it is useless to follow editing links; on a blog, all articles can usually be collected by accessing (for instance) the monthly archives; etc. By formalizing the notion of *Web application* as a hierarchy (individual Web sites, their content management systems, their general category, etc.), we can provide Web crawlers with better guidelines. These descriptions of Web applications and guidelines can be specified by hand, or automatically learned based on the relevance of the discovered Web pages.

Contributions

My PhD student Faheem has started working on intelligent content acquisition in (Faheem 2012). One particular type of Web content where we can have intelligent crawling is Web feeds, that indicate dynamic changes to a Web page. We have explained in (Oita and Senellart 2010) how they can be used for that purpose, and in (Oita and Senellart 2011) we have surveyed the more general problem of determining when a Web page has changed.

Bibliography

Self References

Abdessalem, Talel, M. Lamine Ba, and Pierre Senellart (2011). "A Probabilistic XML Merging Tool". In: *EDBT*. Demonstration. Uppsala, Sweden.

Abiteboul, Serge, Yael Amsterdamer, Daniel Deutch, Tova Milo, and Pierre Senellart (2012). "Finding Optimal Probabilistic Generators for XML Collections". In: *ICDT*. Berlin, Germany.

Abiteboul, Serge, Yael Amsterdamer, Tova Milo, and Pierre Senellart (2012). "Auto-Completion Learning for XML". In: *SIGMOD*. Demonstration. Scottsdale, USA.

Abiteboul, Serge, T-H. Hubert Chan, Evgeny Kharlamov, Werner Nutt, and Pierre Senellart (2010). "Aggregate Queries for Discrete and Continuous Probabilistic XML". In: *ICDT*. Lausanne, Switzerland.

— (2011). "Capturing Continuous Data and Answering Aggregate Queries in Probabilistic XML". In: *ACM Transactions on Database Systems* 36.4.

Abiteboul, Serge, Benny Kimelfeld, Yehoshua Sagiv, and Pierre Senellart (2009). "On the Expressiveness of Probabilistic XML Models". In: *VLDB Journal* 18.5.

Abiteboul, Serge, Ioana Manolescu, Marie-Christine Rousset, Philippe Rigaux, and Pierre Senellart (2012). *Web Data Management*. New York, USA: Cambridge University Press.

Abiteboul, Serge and Pierre Senellart (2006). "Querying and Updating Probabilistic Information in XML". In: *EDBT*. Munich, Germany.

Abiteboul, Serge, Pierre Senellart, and Victor Vianu (2012). "The ERC Webdam on Foundations of Web Data Management". In: *WWW*. European project track. Lyon, France.

Ba, M. Lamine, Talel Abdessalem, and Pierre Senellart (2011). "Towards a Version Control Model with Uncertain Data". In: *PIKM*. Glasgow, United Kingdom.

Benedikt, Michael, Tim Furche, Andreas Savvides, and Pierre Senellart (2012). "ProFoUnd: Program-analysis–based Form Understanding". In: *WWW*. Demonstration. Lyon, France.

Benedikt, Michael, Georg Gottlob, and Pierre Senellart (2011). "Determining Relevance of Accesses at Runtime". In: *PODS*. Athens, Greece.

Benedikt, Michael, Evgeny Kharlamov, Dan Olteanu, and Pierre Senellart (2010). "Probabilistic XML via Markov Chains". In: *Proceedings of the VLDB Endowment* 3.1.

Blondel, Vincent D., Anahí Gajardo, Maureen Heymans, Pierre Senellart, and Paul Van Dooren (2004). "A measure of similarity between graph vertices: applications to synonym extraction and Web searching". In: *SIAM Review* 46.4.

Galland, Alban, Serge Abiteboul, Amélie Marian, and Pierre Senellart (2010). "Corroborating Information from Disagreeing Views". In: *WSDM*. New York, USA.

Gottlob, Georg and Pierre Senellart (2010). "Schema Mapping Discovery from Data Instances". In: *Journal of the ACM* 57.2.

Kharlamov, Evgeny, Werner Nutt, and Pierre Senellart (2010). "Updating Probabilistic XML". In: *Updates in XML*. Lausanne, Switzerland.

— (2011). "Value Joins are Expensive over (Probabilistic) XML". In: *LID*. Uppsala, Sweden.

Kharlamov, Evgeny and Pierre Senellart (2011). "Modeling, Querying, and Mining Uncertain XML Data". In: *XML Data Mining: Models, Methods, and Applications*. Ed. by Andrea Tagarelli. IGI Global.

Kimelfeld, Benny and Pierre Senellart (2012). "Probabilistic XML: Models and Complexity". In: *Advances in Probabilistic Databases for Uncertain Information Management*. Ed. by Zongmin Ma. To appear. Springer-Verlag.

Oita, Marilena and Pierre Senellart (2010). "Archiving Data Objects Using Web Feeds". In: *IWAW*. Vienna, Austria.

— (2011). "Deriving Dynamics of Web Pages: A Survey". In: *TWAW*. Hyderabad, India.

Ollivier, Yann and Pierre Senellart (2007). "Finding Related Pages Using Green Measures: An Illustration with Wikipedia". In: *AAAI*. Vancouver, Canada.

Senellart, Pierre (2007). "Comprendre le Web caché. Understanding the Hidden Web." PhD thesis. Orsay, France: Université Paris XI.

Senellart, Pierre and Serge Abiteboul (2007). "On the Complexity of Managing Probabilistic XML Data". In: *PODS*. Beijing, China.

Senellart, Pierre and Vincent D. Blondel (2008). "Automatic discovery of similar words". In: *Survey of Text Mining II: Clustering, Classification and Retrieval*. Ed. by Michael W. Berry and Malu Castellanos. Springer-Verlag.

Senellart, Pierre and Georg Gottlob (2008). "On the Complexity of Deriving Schema Mappings from Database Instances". In: *PODS*. Vancouver, Canada.

Senellart, Pierre, Avin Mittal, Daniel Muschick, Rémi Gilleron, and Marc Tommasi (2008). "Automatic Wrapper Induction from Hidden-Web Sources with Domain Knowledge". In: *WIDM*. Napa, USA.

Senellart, Pierre and Asma Souihli (2011). "ProApproX: A Lightweight Approximation Query Processor over Probabilistic Trees". In: *SIGMOD*. Demonstration. Athens, Greece.

Suchanek, Fabian M., Serge Abiteboul, and Pierre Senellart (2011). "PARIS: Probabilistic Alignment of Relations, Instances, and Schema". In: *Proceedings of the VLDB Endowment* 5.3.

Vazirgiannis, Michalis, Dimitris Drosos, Pierre Senellart, and Akrivi Vlachou (2008). "Web Page Rank Prediction with Markov Models". In: *WWW*. Poster. Beijing, China.

External References

Afrati, Foto N., Rada Chirkova, Manolis Gergatsoulis, Benny Kimelfeld, Vassia Pavlaki, and Yehoshua Sagiv (2009). "On rewriting XPath queries using views". In: *EDBT*.

Amer-Yahia, S., S. Cho, L. V. S. Lakshmanan, and D. Srivastava (2001). "Minimization of Tree Pattern Queries". In: *SIGMOD*.

Bex, Geert Jan, Frank Neven, and Stijn Vansummeren (2007). "Inferring XML Schema Definitions from XML Data". In: *VLDB*.

Brin, Sergey and Lawrence Page (1998). "The Anatomy of a Large-Scale Hypertextual Web Search Engine". In: *Computer Networks* 30.1-7.

Bruno, N., N. Koudas, and D. Srivastava (2002). "Holistic twig joins: optimal XML pattern matching". In: *SIGMOD*.

Buneman, Peter, Martin Grohe, and Christoph Koch (2003). "Path Queries on Compressed XML". In: *VLDB*.

Cautis, Bogdan, Alin Deutsch, and Nicola Onose (2008). "XPath Rewriting Using Multiple Views: Achieving Completeness and Efficiency". In: *WebDB*.

Cautis, Bogdan and Evgeny Kharlamov (2011). "Challenges for View-Based Query Answering over Probabilistic XML". In: *AMW*.

Chang, L., J. Xu Yu, and L. Qin (2009). "Query ranking in probabilistic XML data". In: *EDBT*.

Cheng, Reynold, Sarvjeet Singh, and Sunil Prabhakar (2005). "U-DBMS: A Database System for Managing Constantly-Evolving Data". In: *VLDB*.

Cohen, Sara and Benny Kimelfeld (2010). "Querying Parse Trees of Stochastic Context-Free Grammars". In: *ICDT*.

Cohen, Sara, Benny Kimelfeld, and Yehoshua Sagiv (2009a). "Incorporating constraints in probabilistic XML". In: *ACM Transactions on Database Systems* 34.3.

— (2009b). "Running tree automata on probabilistic XML". In: *PODS*.

Dalvi, Nilesh, Chrisopher Ré, and Dan Suciu (2009). "Probabilistic Databases: Diamonds in the Dirt". In: *Communications of the ACM* 52.7.

Dalvi, Nilesh N. and Dan Suciu (2007a). "Efficient query evaluation on probabilistic databases". In: *VLDB Journal* 16.4.

— (2007b). "The dichotomy of conjunctive queries on probabilistic structures". In: *PODS*.

Detwiler, Landon, Wolfgang Gatterbauer, Brenton Louie, Dan Suciu, and Peter Tarczy-Hornoch (2009). "Integrating and Ranking Uncertain Scientific Data". In: *ICDE*.

Deutch, Daniel and Tova Milo (2012). "Mob Data Sourcing". In: *SIGMOD*. Tutorial.

Doner, John (1970). "Tree Acceptors and Some of Their Applications". In: *Journal of Computer Systems and Science* 4.5.

Downey, R. G. and M. R. Fellows (1999). *Parameterized Complexity*. Monographs in Computer Science. Springer.

Etessami, Kousha and Mihalis Yannakakis (2009). "Recursive Markov chains, stochastic grammars, and monotone systems of nonlinear equations". In: *Journal of the ACM* 56.1.

Fagin, Ronald, Benny Kimelfeld, and Phokion Kolaitis (2010). "Probabilistic Data Exchange". In: *ICDT*.

Faheem, Muhammad (2012). "Intelligent crawling of Web applications for Web archiving". In: *PhD Symposium of WWW*.

Flum, Jörg and Martin Grohe (2006). *Parameterized Complexity Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer.

Frick, Markus and Martin Grohe (2004). "The complexity of first-order and monadic second-order logic revisited". In: *Annals of Pure and Applied Logic* 130.1-3.

Galindo, José, Angelica Urrutia, and Mario Piattini (2005). *Fuzzy Databases: Modeling, Design And Implementation*. IGI Global.

Green, Todd J., Gregory Karvounarakis, and Val Tannen (2007). "Provenance semirings". In: *PODS*.

Green, Todd J. and Val Tannen (2006). "Models for Incomplete and Probabilistic Information". In: *EDBT Workshops*.

Henriksen, J.G., J. Jensen, M. Jrgensen, N. Klarlund, B. Paige, T. Rauhe, and A. Sandholm (1995). "Mona: Monadic Second-order logic in practice". In: *TACAS*.

Hollander, Emiel and Maurice van Keulen (2010). "Storing and Querying Probabilistic XML Using a Probabilistic Relational DBMS". In: *MUD*.

Huang, Jiewen, Lyublena Antova, Christoph Koch, and Dan Olteanu (2009). "MayBMS: a probabilistic database management system". In: *SIGMOD*.

Hung, Edward, Lise Getoor, and V. S. Subrahmanian (2003). "PXML: A Probabilistic Semistructured Data Model and Algebra". In: *ICDE*.

Imieliński, Tomasz and Witold Lipski Jr. (1984). "Incomplete Information in Relational Databases". In: *Journal of the ACM* 31.4.

Jampani, Ravi, Fei Xu, Mingxi Wu, Luis Leopoldo Perez, Christopher M. Jermaine, and Peter J. Haas (2008). "MCDB: a Monte Carlo approach to managing uncertain data". In: *SIGMOD*.

Jousse, Florent, Rémi Gilleron, Isabelle Tellier, and Marc Tommasi (2006). "Conditional Random Fields for XML trees". In: *ECML Workshop on Mining and Learning in Graphs*.

Karp, Richard M., Michael Luby, and Neal Madras (1989). "Monte-Carlo Approximation Algorithms for Enumeration Problems". In: *Journal of Algorithms* 10.3.

Keulen, Maurice van and Ander de Keijzer (2009). "Qualitative effects of knowledge rules and user feedback in probabilistic data integration". In: *VLDB Journal* 18.5.

Keulen, Maurice van, Ander de Keijzer, and Wouter Alink (2005). "A Probabilistic XML Approach to Data Integration". In: *ICDE*.

Kimelfeld, B., Y. Kosharovsky, and Y. Sagiv (2008). "Query Efficiency in Probabilistic XML Models". In: *SIGMOD*.

Kimelfeld, Benny, Yuri Kosharovsky, and Yehoshua Sagiv (2009). "Query evaluation over probabilistic XML". In: *VLDB Journal* 18.5.

Kimelfeld, Benny and Yehoshua Sagiv (2007). "Matching Twigs in Probabilistic XML". In: *VLDB*.

Kleinberg, Jon M. (1999). "Authoritative Sources in a Hyperlinked Environment". In: *Journal of the ACM* 46.5.

Lafferty, John, Andrew McCallum, and Fernando Pereira (2001). "Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data". In: *ICML*.

Li, J., C. Liu, R. Zhou, and W. Wang (2011). "Top-k keyword search over probabilistic XML data". In: *ICDE*.

Li, Te, Qihong Shao, and Yi Chen (2006). "PEPX: a query-friendly probabilistic XML database". In: *MUD*.

Li, Y., G. Wang, J. Xin, E. Zhang, and Z. Qiu (2009). "Holistically Twig Matching in Probabilistic XML". In: *ICDE*.

Manning, Christopher D. and Hinrich Schutze (1999). *Foundations of Statistical NLP*. MIT Press.

Meyer, Albert R. (1975). "Weak monadic second-order theory of successor is not elementary recursive". In: *Logic Colloquium*. Lecture Notes in Mathematics 453.

Neven, F. and T. Schwentick (2002). "Query automata over finite trees". In: *Theoretical Computer Science* 275.1-2.

Nierman, Andrew and H. V. Jagadish (2002). "ProTDB: Probabilistic Data in XML". In: *VLDB*.

Olteanu, Dan, Jiewen Huang, and Christoph Koch (2009). "SPROUT: Lazy vs. Eager Query Plans for Tuple-Independent Probabilistic Databases". In: *ICDE*.

Robertson, Neil and P. D. Seymour (1984). "Graph minors. III. Planar tree-width". In: *Journal of Combinatorial Theory, Series B* 36.1.

Sen, Prithviraj, Amol Deshpande, and Lise Getoor (2009). "PrDB: managing and exploiting rich correlations in probabilistic databases". In: *VLDB Journal* 18.5.

Souihli, Asma (2011). "Efficient Query Evaluation Over Probabilistic XML With Long-Distance Dependencies". In: *EDBT/ICDT PhD Workshop*.

Suciu, Dan, Dan Olteanu, Christopher Ré, and Christoph Koch (2011). *Probabilistic Databases*. Morgan & Claypool.

Thatcher, James W. and Jesse B. Wright (1968). "Generalized Finite Automata Theory with an Application to a Decision Problem of Second-Order Logic". In: *Mathematical Systems Theory* 2.1.

Toda, S. and M. Ogiwara (1992). "Counting Classes are at Least as Hard as the Polynomial-Time Hierarchy". In: *SIAM Journal on Computing* 21.2.

Valiant, L. G. (1979). "The Complexity of Computing the Permanent". In: *Theoretical Computer Science* 8.

Vardi, Moshe Y. (1982). "The Complexity of Relational Query Languages (Extended Abstract)". In: *STOC*.

Veldman, Irma, Ander de Keijzer, and Maurice van Keulen (2009). "Compression of Probabilistic XML Documents". In: *SUM*.

Widom, Jennifer (2005). "Trio: A System for Integrated Management of Data, Accuracy, and Lineage". In: *CIDR*.

Xu, Wanhong and Z. Özsoyoglu (2005). "Rewriting XPath Queries Using Materialized Views". In: *VLDB*.

Xu, Y. and Y. Papakonstantinou (2005). "Efficient Keyword Search for Smallest LCAs in XML Databases". In: *SIGMOD*.

Zadeh, Lotfi A. (1986). "A Simple View of the Dempster-Shafer Theory of Evidence and Its Implication for the Rule of Combination". In: *AI Magazine* 7.2.