Probabilistic XML: Survey and Challenges

Pierre Senellart

Groupe de travail Dahu
13 January 2010
Outline

1 Motivation

2 Probabilistic XML Survey

3 Challenges
Uncertain data

Numerous sources of uncertain data:

- Measurement errors
- Data integration from contradicting sources
- Imprecise mappings between heterogeneous schemata
- Imprecise automatic process (information extraction, natural language processing, etc.)
- Imperfect human judgment
Managing this imprecision

Objective

Not to pretend this imprecision does not exist, and manage it as rigorously as possible throughout a long, automatic and human, potentially complex, process.

Especially:

- Use probabilities to represent the confidence in the data
- Query data and retrieve probabilistic results
- Allow adding, deleting, modifying data in a probabilistic way
- (If possible) Keep throughout the process lineage/provenance information, so as to ensure traceability
Managing this imprecision

Objective

Not to pretend this imprecision does not exist, and manage it as rigorously as possible throughout a long, automatic and human, potentially complex, process.

Especially:

- Use **probabilities** to represent the confidence in the data
- Query data and retrieve **probabilistic** results
- Allow adding, deleting, modifying data in a **probabilistic** way
- (If possible) Keep throughout the process **lineage/provenance** information, so as to ensure **traceability**
Why XML?

- Extensive literature about probabilistic relational databases [DRS09, Wid05, Koc09]
- Different typical querying languages: conjunctive queries vs tree-pattern queries (possibly with joins)
- Cases where a tree-like model might be appropriate:
  - No schema or few constraints on the schema
  - Independent modules annotating freely a content warehouse
  - Inherently tree-like data (e.g., mailing lists, parse trees) with naturally occurring queries involving the descendant axis

Remark

Some results can be transferred from one model to the other. In other cases, connection much trickier! (See later.)
1 Motivation

2 Probabilistic XML Survey
   - Models
   - Querying
   - Other Problems of Interest

3 Challenges
1 Motivation

2 Probabilistic XML Survey
   - Models
   - Querying
   - Other Problems of Interest

3 Challenges
Trees and possible worlds

Unordered data trees

Sample space: Set of all such data trees.

Probabilistic XML database: (Succinct) representation of a discrete probability distribution over this sample space (= a set of possible worlds).
Trees and possible worlds

Unordered data trees

Sample space: Set of all such data trees.

Probabilistic XML database: (Succinct) representation of a discrete probability distribution over this sample space (= a set of possible worlds).
Unordered data trees

Sample space: Set of all such data trees.

Probabilistic XML database: (Succinct) representation of a discrete probability distribution over this sample space (= a set of possible worlds).
Local dependencies \[NJ02, KKS08\]

- **Tree with ordinary** (circles) and **distributional** (rectangles) nodes

- Distributional nodes specify how their children can be **randomly selected** (here, independently or in a mutually exclusive way)

- **Possible-world semantics**: every possible selection of children of distributional nodes, with associated probability

- No long-distance probabilistic dependencies in the tree!
Types of distributional nodes [KKS08, AKSS09]

- **det**: all children of the node are deterministically selected.
- **ind**: children of the node are chosen independently of one another, according to their probabilities.
- **mux**: children of the node are chosen in a mutually exclusive way, depending of their probabilities, that must sum up to 1 or less.
- **exp**: the distribution of all possible choices of children is explicitly given: each subset of the set of the children is associated with a probability, these probabilities summing up to 1.

**Remark**

Clearly, det is a particular case of ind, and mux is a particular case of exp.
Arbitrary dependencies: event conjunctions [AS06]

Conjunctions of independent events on each node of the tree [IL84]
Expresses arbitrarily complex dependencies
Both ind and mux can be seen as particular cases (but not exp!) [AKSS09]
Arbitrary dependencies: event conjunctions \cite{AS06}

\begin{itemize}
  \item Conjunctions of independent events on each node of the tree \cite{IL84}
  \item Expresses arbitrarily complex dependencies
  \item Both \textit{ind} and \textit{mux} can be seen as particular cases (but not \textit{exp}) \cite{AKSS09}
\end{itemize}

\begin{table}[h]
\centering
\begin{tabular}{ll}
\hline
Event & Prob. \\
\hline
$w_1$ & 0.8 \\
$w_2$ & 0.7 \\
\hline
\end{tabular}
\end{table}
Events and lineage

- Event variables: can represent the **provenance** of data
- Typically:
  1. At each (probabilistic) update, a **new** event variable is introduced
  2. Query results are given with probabilities, but also with the **lineage** of the query [FGT08]
- Allow to keep **track**, with no additional cost, of the provenance of data!
Previously studied XML models

ProTDB [NJ02]  ind + mux

Probabilistic XML [vKdKA05]  mux + det, with alternation between the two kinds of nodes

SP trees [AS06], PEPX [LSC06]  ind without hierarchies of distributional nodes

PXML [HGS03]  exp without hierarchies, extended to graphs

Probabilistic interval XML [HGS07]  exp without hierarchies, when intervals are collapsed into points

Prob-trees [AS06, SA07]  Event conjunctions
Expressiveness

Theorem ([AS06, KKS08, AKSS09])

1. ind alone, or mux alone, are not a complete representation system.
2. det + mux is enough to have full expressive power. Consequently, ind + mux, exp alone, or event conjunctions, have full expressive power.
3. Hierarchies (allowing a distributional node below another distributional node) are important.
Tractable reductions between models [KKS08, AKSS09]

\[ \text{det} + \text{mux} = \text{ind} + \text{mux} \]

\[ \text{exp} + \text{Event conjunctions} \]

\[ \text{Event conjunctions} \]
Tractable reductions between models [KKS08, AKSS09]

\[ \text{exp + Event conjunctions} \]

\[ \text{exp} \rightarrow \text{Event conjunctions} \]

\[ \text{det + mux = ind + mux} \]
1 Motivation

2 Probabilistic XML Survey
   - Models
   - Querying
   - Other Problems of Interest

3 Challenges
Semantics of a (Boolean) query = probability:

1. Generate all possible worlds of a given probabilistic document

2. In each world, evaluate the query

3. Add up the probabilities of the worlds that make the query true

EXPTIME algorithm! Can we do better, i.e., can we apply directly the algorithm on the probabilistic document?

We shall talk about data complexity of query answering.
Semantics of a (Boolean) query = probability:

1. Generate all possible worlds of a given probabilistic document (possibly exponentially many)
2. In each world, evaluate the query
3. Add up the probabilities of the worlds that make the query true

EXPTIME algorithm! Can we do better, i.e., can we apply directly the algorithm on the probabilistic document?

We shall talk about data complexity of query answering.
Boolean query languages on trees

Single-path queries (SP) /A//B/C (no branching)

Tree-pattern queries (TP) /A[C/D]//B

Tree-pattern queries with joins (TPJ) for $x$ in $\text{doc}/A/C/D$

return $\text{doc}/A//B[.=$x$]$

Monadic second-order queries (MSO) generalization of TP, do not cover TPJ unless the size of the alphabet is bounded
The \#P and FP\#P complexity classes

- A (counting) problem is in \#P if there is a PTIME non-deterministic Turing machine whose number of accepting paths, given as input the input of the problem, is the output of the problem.

- A problem is \#P-hard if any \#P problem can be PTIME-reduced to it (via a Turing reduction). \#2DNF, the problem of counting the number of assignments satisfying a formula in 2-DNF, is \#P-complete.

- A (computation) problem is in FP\#P if it is computable by a PTIME Turing machine with access to a \#P oracle.

- A problem is FP\#P-hard if any FP\#P problem can be PTIME-reduced to it (via a Turing reduction). Equivalently, a computation problem is FP\#P-hard if it is \#P-hard.
The $\#P$ and $FP^{\#P}$ complexity classes

- A (counting) problem is in $\#P$ if there is a PTIME non-deterministic Turing machine whose number of accepting paths, given as input the input of the problem, is the output of the problem.

- A problem is $\#P$-hard if any $\#P$ problem can be PTIME-reduced to it (via a Turing reduction). $\#2\text{DNF}$, the problem of counting the number of assignments satisfying a formula in 2-DNF, is $\#P$-complete.

- A (computation) problem is in $FP^{\#P}$ if it is computable by a PTIME Turing machine with access to a $\#P$ oracle.

- A problem is $FP^{\#P}$-hard if any $FP^{\#P}$ problem can be PTIME-reduced to it (via a Turing reduction). Equivalently, a computation problem is $FP^{\#P}$-hard if it is $\#P$-hard.
### Complexity of query evaluation

<table>
<thead>
<tr>
<th></th>
<th>Local dependencies</th>
<th>Arbitrary dependencies</th>
</tr>
</thead>
<tbody>
<tr>
<td>SP</td>
<td>PTIME</td>
<td>FP(^#P)-complete [KKS08]</td>
</tr>
<tr>
<td>TP</td>
<td>PTIME [KS07, KKS08, KKS09]</td>
<td>FP(^#P)-complete</td>
</tr>
<tr>
<td>TPJ</td>
<td>FP(^#P)-complete</td>
<td>FP(^#P)-complete</td>
</tr>
<tr>
<td>MSO</td>
<td>PTIME [CKS09]</td>
<td>FP(^#P)-complete</td>
</tr>
</tbody>
</table>

**Remark**

Project-free queries are tractable with arbitrary dependencies. \[SA07\]
### Complexity of query evaluation

<table>
<thead>
<tr>
<th>Local dependencies</th>
<th>Arbitrary dependencies</th>
</tr>
</thead>
<tbody>
<tr>
<td>TP</td>
<td>PTIME [KS07, KKS08, KKS09]</td>
</tr>
<tr>
<td>TPJ</td>
<td>FP#P-complete</td>
</tr>
</tbody>
</table>

Remark: Project-free queries are tractable with arbitrary dependencies. [SA07]
Bottom-up dynamic programming algorithm.
Query: /A//B

\[
\begin{array}{c|c|c|c|c|c|c}
  & A_1 & \text{ind}_2 & \text{mux}_3 & B_4 & C_5 & B_6 \\
\hline
/B & 1 & 0 & 1 \\
//B & 1 & 0 & 1 \\
/A//B & 0 & 0 & 0 \\
\end{array}
\]

- \text{mux} convex sum
- \text{ind} inclusion-exclusion
- \text{ordinary} inclusion-exclusion
Bottom-up dynamic programming algorithm.
Query: /A//B

<table>
<thead>
<tr>
<th></th>
<th>$A_1$</th>
<th>$\text{ind}_2$</th>
<th>$\text{mux}_3$</th>
<th>$B_4$</th>
<th>$C_5$</th>
<th>$B_6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>/B</td>
<td>0.3</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>//B</td>
<td>0.3</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>/A//B</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

mux: convex sum
ind: inclusion-exclusion
ordinary: inclusion-exclusion
Bottom-up dynamic programming algorithm.
Query: /A//B

\[
\begin{array}{cccccc}
\text{event} & A_1 & \text{ind}_2 & \text{mux}_3 & B_4 & C_5 & B_6 \\
\text{/B} & 0.696 & 0.3 & 1 & 0 & 1 \\
\text{///B} & 0.696 & 0.3 & 1 & 0 & 1 \\
\text{/A///B} & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]

- \text{mux} convex sum
- \text{ind} inclusion-exclusion
- \text{ordinary} inclusion-exclusion

\[
\Pr(\text{\text{ind}_2} \models /B) = 1 - (1 - 0.8 \times \Pr(\text{\text{mux}_3} \models /B)) \times (1 - 0.6 \times \Pr(\text{B}_6 \models /B))
\]
\[
= 1 - (1 - 0.8 \times 0.3) \times (1 - 0.6) = 0.696
\]
Bottom-up dynamic programming algorithm.
Query: /A//B

<table>
<thead>
<tr>
<th></th>
<th>$A_1$</th>
<th>ind$_2$</th>
<th>mux$_3$</th>
<th>$B_4$</th>
<th>$C_5$</th>
<th>$B_6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>/B</td>
<td>0</td>
<td>0.696</td>
<td>0.3</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>//B</td>
<td>0.696</td>
<td>0.696</td>
<td>0.3</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>/A//B</td>
<td>0.696</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

mux convex sum
ind inclusion-exclusion
ordinary inclusion-exclusion
Complications \cite{KS07, KKS08, KKS09}

General case:

- Branching patterns: need to consider all \textit{conjunctions of (possibly negated) subpatterns} of a pattern (exponentially many!)
- Works also with $\exp$
- Number of optimizations possible
- **Bottomline:** it works because bottom-up evaluation is possible
- **Generalization** \cite{CKS09}: MSO queries can be converted (non efficiently) into bottom-up tree automata, therefore MSO is also tractable
Reduction from $\#2\text{DNF}$. Example: $\varphi = xy \lor x \neg z \lor yz$. 

\[
\text{mux} \quad \text{det} \\
\ell \quad 1 \\
0 \\
\text{mux} \quad \text{det} \\
\ell \quad 0 \\
0 \\
\text{mux} \quad \text{det} \\
r \quad 2 \\
r \quad 1
\]
In some sense, all variants of probabilistic XML are still tractable:

- **Additive FPRAS** for $\exp +$ event conjunctions: simple Monte Carlo sampling
- ... but additive approximation is not good enough for low probabilities
- **Multiplicative FPRAS** for $\exp +$ event conjunctions: minor rewriting then biased Monte Carlo
Aggregate Queries [CKS08, ACK+10]

Aggregate Queries: sum, count, avg, countd, min, max, etc. Distributions? Possible values? Expected value?

Summary of results

- Computing **HAVING** queries is **tractable** for count, max, min, ratio.
- Computing expected values of sum and count **tractable** with arbitrary dependencies. Everything else **intractable**.
- Computing expected values of every of these aggregate functions is **tractable** with local dependencies.
- Computing distributions and possible values is **tractable** for count, min, max, **intractable** for the others.

Always possible to approximate query answers with **Monte Carlo sampling**.
Outline

1 Motivation

2 Probabilistic XML Survey
   ■ Models
   ■ Querying
   ■ Other Problems of Interest

3 Challenges
Determining the probability that a probabilistic document with local dependencies matches a schema is tractable (uses the transformation of schemas into bottom-up automata).

Determining the probability that a probabilistic document with arbitrary dependencies matches a schema is intractable.
Updates defined by a query (cf. XUpdate, XQuery Update).
Semantics: for all matches of a query, insert or delete a node in the tree at a place located by the query.

Results

- Most updates are intractable with local dependencies: the result of an update can require an exponentially larger representation size.

- Insertions with a for-all-match semantics are tractable with arbitrary dependencies; deletions are intractable.

- Some insert-if-there-is-a-match operations tractable for local dependencies but not for arbitrary dependencies.
Outline

1 Motivation

2 Probabilistic XML Survey

3 Challenges
Missing complexity results

- Tractable reduction from exp to arbitrary dependencies?
- Tractable reduction from exp to mux + ind?
- Combined complexity results.
Link with probabilistic relational models

Relational case
(Block-independent disjoint model, [DS07])

- Some conjunctive queries are PTIME
- Others are \#P-hard
- Complex conditions to separate the two

XML case (Local dependencies)

- Tree pattern queries are PTIME
- Tree pattern queries with (non-trivial) joins are \#P-hard

Why does the XML case seem simpler?
Is there some insight to be gained from one case to the other?
Translating XML data and queries to the relational case yields queries with self-joins, a less well-understood setting
Link with probabilistic relational models

Relational case
(Block-independent disjoint model, [DS07])

- Some conjunctive queries are \textbf{PTIME}
- Others are \textbf{#P-hard}
- Complex conditions to separate the two

XML case (Local dependencies)

- Tree pattern queries are \textbf{PTIME}
- Tree pattern queries with (non-trivial) joins are \textbf{#P-hard}

- Why does the XML case seem simpler?
- Is there some insight to be gained from one case to the other?
- Translating XML data and queries to the relational case yields queries with self-joins, a less well-understood setting
Continuous probability distributions

- Most probabilistic database models assume discrete probabilistic distributions
- Sensor networks, unknown values: need for continuous distributions! (uniform, Gaussian, Poisson, etc.)
- Some existing works on query answering over continuous distributions [CKP03, DGM⁺04] but no clear semantics
- **Claim:** this is not more difficult than the discrete case, as long as integration/differentiation are easy (symbolically or numerically) for the considered distributions
- Discrete distributions can be modeled as Diracs

Work in progress with U. Bozen-Bolzano [ACK⁺10]
Tractable extensions of the local dependency model

- Arbitrary dependencies: not tractable
- Local dependencies: not practical
- Somewhere in between?
  - What makes the arbitrary dependency model hard?
  - How can the local dependency model be generalized, while remaining tractable?
- And can we go further? cf. XML schemas
  - Trees of unbounded depth
  - Trees of unbounded width
  - Infinite trees?

Work in progress with U. Oxford [BKOS09]
Example: prob. DTDs via rec. Markov chains \cite{BKOS09}

\[
\text{<!ELEMENT directory (person*)>}
\]
\[
\text{<!ELEMENT person (name,phone*)>}
\]

\[D: \text{directory} \]

\[P: \text{person} \]

\[N: \text{name} \]

\[T: \text{phone} \]

On such simple RMCs representing trees, MSO queries are tractable!
But where do probabilities come from?!

- Do the numbers assigned as probabilities in PDBMS really make sense?
- In some cases, sources of “good” probabilities:
  - Statistics
  - Conditional Random Fields [LMP01]
  - Parse Trees of Stochastic Context-Free Grammars [CK10]
  - Uncertain Schema Matching [DHY09, FKK10]
  - Representing a Corpus with a Probabilistic Documents?
- What about the rest? Does it really make sense to model uncertainty with probabilities?
Nothing else than toy systems exist for probabilistic XML

What should it be based upon:
- a probabilistic relational DBMS?
- a native XML DBMS?

Systems issue: distribution, indexing, etc.

And need for a killer application!
- Probabilistic content warehouse?
- Parse trees of natural language sentences?
- Concise representation of a large corpus of XML documents?

PhD started on this topic in October 2009
Merci.
Aggregate queries for discrete and continuous probabilistic xml.
In *Proc. ICDT*, Lausanne, Switzerland, March 2010.

Serge Abiteboul, Benny Kimelfeld, Yehoshua Sagiv, and Pierre Senellart.
On the expressiveness of probabilistic XML models.

Serge Abiteboul and Pierre Senellart.
Querying and updating probabilistic information in XML.
In *Proc. EDBT*, Munich, Germany, March 2006.
Michael Benedikt, Evgeny Kharlamov, Dan Olteanu, and Pierre Senellart.

Sara Cohen and Benny Kimelfeld.
Querying parse trees of stochastic context-free grammars.
In Proc. ICDT, Lausanne, Switzerland, March 2010.

Reynold Cheng, Dmitri V. Kalashnikov, and Sunil Prabhakar.
Evaluating probabilistic queries over imprecise data.


Nilesh Dalvi, Christopher Ré, and Dan Suciu.
Probabilistic databases: Diamonds in the dirt.
*Communications of the ACM*, 52(7), 2009.

Nilesh N. Dalvi and Dan Suciu.
Management of probabilistic data: foundations and challenges.
In *Proc. PODS*, Beijing, China, June 2007.

J. Nathan Foster, Todd J. Green, and Val Tannen.
Annotated XML: queries and provenance.

Ronald Fagin, Benny Kimelfeld, and Phokion Kolaitis.
Probabilistic data exchange.
In *Proc. ICDT*, Lausanne, Switzerland, March 2010.


B. Kimelfeld and Y. Sagiv.  
Matching twigs in probabilistic XML.  

John Lafferty, Andrew McCallum, and Fernando Pereira.  
Conditional Random Fields: Probabilistic models for segmenting and labeling sequence data.  

Te Li, Qihong Shao, and Yi Chen.  

Andrew Nierman and H. V. Jagadish.  
ProTDB: Probabilistic data in XML.  
In *Proc. VLDB*, Hong Kong, China, August 2002.
Pierre Senellart and Serge Abiteboul.
On the complexity of managing probabilistic XML data.
In *Proc. PODS*, Beijing, China, June 2007.

Maurice van Keulen, Ander de Keijzer, and Wouter Alink.
A probabilistic XML approach to data integration.
In *Proc. ICDE*, Tokyo, Japan, April 2005.

Jennifer Widom.
Trio: A system for integrated management of data, accuracy, and lineage.