Value Joins are Expensive over (Probabilistic) XML

E. Kharlamov, W. Nutt, P. Senellart
Outline

Probabilistic XML

Querying Probabilistic XML

The Complexity of Joins

Essential Joins

Conclusion
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
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

Uncertainty modeled here as probabilities
Why XML?

- Extensive literature about probabilistic relational databases [Dalvi et al., 2009, Widom, 2005, Koch, 2009]
- 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
Local dependencies

[Nierman and Jagadish, 2002, Kimelfeld et al., 2008]

- Tree with **ordinary** (circles) and **distributional** (rectangles) nodes
- Distributional nodes specify how their children can be **randomly selected**:
  - **ind** independently of one another;
  - **det** deterministically;
  - **mux** mutually exclusively.
- **Possible-world semantics**: every possible selection of children of distributional nodes, with associated probability
- No long-distance probabilistic dependencies in the tree!
Local dependencies

[Nierman and Jagadish, 2002, Kimelfeld et al., 2008]

- Tree with **ordinary** (circles) and **distributional** (rectangles) nodes
- Distributional nodes specify how their children can be randomly selected:
  - **ind** independently of one another;
  - **det** deterministically;
  - **mux** mutually exclusively.

Focus here on such **local dependencies**

Other **more expressive** (and less tractable) probabilistic XML models exist [Abiteboul et al., 2009, Cohen et al., 2008, Kharlamov et al., 2010, Benedikt et al., 2010]

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

- No long-distance probabilistic dependencies in the tree!
Outline

Probabilistic XML

Querying Probabilistic XML

The Complexity of Joins

Essential Joins

Conclusion
Semantics of queries

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
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! We usually want to do better, i.e., to apply directly the algorithm on the probabilistic document?

Focus on data complexity
Boolean query languages on trees

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

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

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

Monadic second-order queries (MSO) generalization of TP, does not cover TPJ unless the size of the alphabet is bounded

Monadic second-order queries with joins (MSOJ) MSO + SameLabel predicate
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.
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.
Motivating Observation

- **Linear algorithm** for computing the probability of a TP query
  [Kimelfeld and Sagiv, 2007, Kimelfeld et al., 2009] and even of an MSO query
  [Cohen et al., 2009]
- Very simple TPJ queries have **#P-hard complexity** over
  probabilistic XML [Abiteboul et al., 2010]
- **Where is the boundary? How hard** are queries with joins?
- Algorithm to decide whether a query is hard?
Outline

Probabilistic XML

Querying Probabilistic XML

The Complexity of Joins

Essential Joins

Conclusion
The complexity of TPJ

Proposition

The data complexity of TPJ evaluation is:

- $\text{PTIME}$ over XML;
- $\text{FP}^\#P$-complete over probabilistic XML.

Main idea: TPJ on trees is basically the same thing as conjunctive queries on relations.
Proposition

The data complexity of MSOJ evaluation is:

- $\Sigma_k^P$-complete and $\Pi_k^P$-complete over XML for all $k \geq 0$;
- $\#P$-hard over probabilistic XML.

Main idea: MSOJ on trees is basically the same thing as MSO on relations.
Outline

Probabilistic XML

Querying Probabilistic XML

The Complexity of Joins

Essential Joins

Conclusion
Definition
A TPJ (resp., MSOJ) query query is essentially join-free if it is equivalent to a TP (resp., MSO) query.
If a query is not essentially join-free, it is said to have essential joins.

Example
\[
\begin{array}{c}
A \\
\downarrow \\
B \quad \text{is equivalent to} \\
\text{B} \\
\end{array}
\]
Theorem

A TPJ query $q$ is essentially join-free if it is equivalent to the query obtained from $q$ by removing all join conditions.

Main idea: characterization of query containment of TP queries as query evaluation on a representative document, due to [Miklau and Suciu, 2004]
Deciding Essential Joins

Theorem

Deciding essential joins is:

- $\Pi_2^P$-complete for TPJ;
- undecidable for MSOJ.

Main idea: similar construction to the one used in [Deutsch and Tannen, 2001] for $\Pi_2^P$-completeness of TPJ query containment
Theorem
Let $q$ be a TPJ query with a single join. Then:

- if $q$ is essentially join-free, then query evaluation of $q$ over PrXML is $\text{PTIME}$;
- otherwise, it is $\text{FP}^\#P$-complete.
Hardness Proof Idea

Reduction from #2DNF. Example: $\varphi = xy \lor x\overline{z} \lor yz$.
Hardness Proof Idea

Reduction from \#2DNF. Example: \( \varphi = xy \lor x\bar{z} \lor yz \).

General case (arbitrary with a single join, not essentially join-free): replace \( l \) and \( r \) with the necessarily distinguishable paths of the tree leading to the join variables. Quite technical!
Probabilistic XML

Querying Probabilistic XML

The Complexity of Joins

Essential Joins

Conclusion
In Brief

- Join-free queries: everything is linear

- Complexity of join queries:

<table>
<thead>
<tr>
<th></th>
<th>TPJ</th>
<th>MSOJ</th>
</tr>
</thead>
<tbody>
<tr>
<td>XML</td>
<td>PTIME</td>
<td>$\Sigma^P_k$-complete, $\Pi^P_k$-complete $\forall k \geq 0$</td>
</tr>
<tr>
<td>PrXML</td>
<td>FP$#^P$-complete</td>
<td>$#P$-hard, in FPSPACE</td>
</tr>
</tbody>
</table>

- Deciding essential joins
  - can be done in $\Pi^P_2$ for TPJ
  - is undecidable for MSOJ

- Being essentially join-free is the tractability criterion for TPJ queries with a single join

See combined complexity results in the paper.
Open Problems

- A number of complexity gaps to fill in
- Extension of the dichotomy result to arbitrary TPJ queries
- Approximation techniques (A. Souihli’s talk at the PhD Workshop in 30min)
Open Problems

- A number of complexity gaps to fill in
- Extension of the dichotomy result to arbitrary TPJ queries
- Approximation techniques (A. Souihli’s talk at the PhD Workshop in 30min)
- Investigating the connection with the (much more complicated) dichotomy of conjunctive queries over relational data [Dalvi and Suciu, 2007]
- Things are easier over trees because of the structure of the data; what about bounded tree-width relations?
- Joins are correlation in the query. What about data correlations (long-distance dependencies)?
Merci.


