The Possibility Problem for Probabilistic XML

Antoine Amarilli
Télécom ParisTech; Institut Mines-Télécom; CNRS LTCI

Abstract. We consider the *possibility problem* of determining if a document is a possible world of a probabilistic document, in the setting of probabilistic XML. This basic question is a special case of query answering or tree automata evaluation, but it has specific practical uses, such as checking whether an user-provided probabilistic document outcome is possible or sufficiently likely.
In this paper, we study the complexity of the possibility problem for probabilistic XML models of varying expressiveness. We show that the decision problem is often tractable in the absence of long-distance dependencies, but that its computation variant is intractable on unordered documents. We also introduce an *explicit matches* variant to generalize practical situations where node labels are unambiguous; this ensures tractability of the possibility problem, even under long-distance dependencies, provided event conjunctions are disallowed. Our results entirely classify the tractability boundary over all considered problem variants.

1 Introduction

Probabilistic representations are a way to represent incomplete knowledge through a concise description of a large set of possible worlds annotated with their probability. Such models can then be used, e.g., to run a query efficiently over all possible worlds and determine the overall probability that the query holds. Probabilistic representations have been successfully used both for the relational model [11] and for XML documents [9].

Many problems, such as query answering [8], have been studied over such representations; however, to our knowledge, the *possibility problem* (POSS) has not been specifically studied: given a probabilistic document D and a deterministic document W, decide if W is a possible world of D, and optionally compute its probability according to D. This can be asked both of relational and XML probabilistic representations, but we focus on XML documents[1] because they pose many challenges: they are hierarchical so some probabilistic choices appear dependent[2]; documents may be ordered; bag semantics must be used to count multiple sibling nodes with the same label. In addition, in the XML setting, the POSS problem is a natural question that arises in practical scenarios.

As a first example, when using probabilistic XML to represent a set D of possible versions [4] of an XML document, one may want to determine if a version W, obtained from a user or from an external source, is one of the known possible versions represented as a probabilistic XML document D. For instance, assume that a probabilistic XML version control system asks a user to resolve a conflict [3], whose uncertain set of possible outcomes is represented by D. When the user provides a candidate merge W, the system must then check if the document W is indeed a possible way to solve the conflict. This

[1] Translations between the probabilistic relational and XML models [2] can be used to translate our results to complexity bounds for the POSS problem on probabilistic relational databases.
[2] In fact, we will see that our hardness results always hold even for shallow documents.

may be hard to determine, because D may, in general, have many ways to generate W, through a possibly intractable number of different valuations of its uncertainty events.

As a second practical example, assume that a user is editing a probabilistic XML document D. The user notices that choosing a certain valuation of the probabilistic events yields a certain deterministic document W, and asks whether the same document could have been obtained by making different choices. Indeed, maybe W is considered improbable under D following this particular valuation, but is likely overall because the same document can be obtained through different choices. What is the probability, over all valuations, of the user's chosen outcome W according to D?

On the face of it, POSS seems related to query evaluation: we wish to evaluate on D a query q_W which is, informally, "is the input document exactly W"? However, there are three reasons why query evaluation cannot give good complexity bounds for POSS. First, because q_W depends on the possibly large W, we are not performing query answering for a *fixed* query, so we can only use the unfavorable *combined complexity* bounds where both the input document D and the query q_W are part of the input. Second, because we want to obtain *exactly* W, the match of q_W should never map two query variables to the same node of D, so the query language must allow inequalities on node identifiers. Third, once again because we require an exact match, we need to assert the *absence* of the nodes which are not in W, so we need negation in the language. To our knowledge, then, the only upper bound for POSS from query answering is the combined complexity bound for the (expressive) *monadic second-order logic over trees* whose evaluation on deterministic (not even probabilistic) XML trees is already PSPACE-hard [10].

A second related approach is that of tree automata on probabilistic XML documents. Indeed, we can encode the possible world W to a deterministic tree automaton A_W and compute the probability that A_W accepts the probabilistic document D. The decision and computation variants of POSS under local uncertainty models are thus special cases of the "relevancy" and "p-acceptance" problems of [6]. However, their work only considers *ordered* trees, and an unordered W cannot easily be translated to their deterministic tree automata, because of possible label ambiguity: we cannot impose an arbitrary order on D and W, as this also chooses how nodes must be disambiguated. In fact, we will show that POSS is hard in some settings that are tractable for ordered documents.

This paper specifically focuses on the POSS problem to study the precise complexity of its different formulations. Our probabilistic XML representation is the PrXML model of [9], noting that some results are known for the POSS problem (called the "membership problem") in the incomparable and substantially different "open-world" incomplete XML model of [5] (whose documents have an infinite set of possible worlds, instead of a possibly exponential but finite set as in PrXML).

We start by defining the required preliminaries in Section 2 and the different variants of POSS in Section 3, establishing its overall NP-completeness and reviewing the results of [6]. We then study local uncertainty models in Section 4 and show that the absence of order impacts tractability, with a different picture for the decision and computation variants of POSS. Last, in Section 5, we show that POSS can be made tractable under long-distance event correlations, by disallowing event conjunctions and imposing an "explicit matches" condition which generalizes, e.g., unique node labels. We then conclude in Section 6. For lack of space, all proofs are deferred to the extended version [1].

2 Preliminaries

We start by formally defining XML documents and probability distributions over them:

Definition 1. *An* unordered XML document *is an unordered tree whose nodes carry a label from a set Λ of labels.* Ordered *XML documents are defined in the same way but with ordered trees, that is, there is a total order over the children of every node.*

A probability distribution *is a function $\mathcal{P}$ mapping every XML document x from a finite set* $\mathrm{supp}(\mathcal{P})$ *to a rational number $\mathcal{P}(x)$, its* probability *according to $\mathcal{P}$, with the condition that $\sum_{D \in \mathrm{supp}(\mathcal{P})} \mathcal{P}(D) = 1$. For any $x \notin \mathrm{supp}(\mathcal{P})$ we write $\mathcal{P}(x) = 0$.*

As it is unwieldy to manipulate explicit probability distributions over large sets of documents, we use the language of probabilistic XML [9] to write extended XML documents (with so-called *probabilistic nodes*) and give them a semantics which is a (possibly exponentially larger) probability distribution over XML documents.

Definition 2. *A* PrXML probabilistic XML document *D is an XML document over $\Lambda \sqcup \{\mathsf{det}, \mathsf{ind}, \mathsf{mux}, \mathsf{cie}, \mathsf{fie}\}$, where every edge from a* mux *or* ind *node to a child node is labeled with some rational number*[3] *$0 < x < 1$ (the sum of the labels of the children of every* mux *node being ≤ 1), and every edge from a* cie *(resp.* fie*) node to a child node is labeled with a conjunction (resp. a Boolean formula) of* events *from a set E of events (and their negations), with a mapping $\pi : E \to [0,1]$ attributing a rational probability to every event. The nodes with labels from Λ are called* regular *nodes, by opposition to* probabilistic *nodes. We assume that the root of D is a regular node.*

For any subset $\mathcal{L} \subseteq \{\mathsf{det}, \mathsf{ind}, \mathsf{mux}, \mathsf{cie}, \mathsf{fie}\}$, we call $\mathsf{PrXML}^{\mathcal{L}}$ the language of probabilistic XML documents containing only nodes with labels in $\Lambda \sqcup \mathcal{L}$.

The semantics of a PrXML document D is the probability distribution over XML documents defined by the following sampling process (see [9] for more details):

Definition 3. *A deterministic XML document W is obtained from a* PrXML *document D as follows. First, choose a valuation $\nu : E \to \{\mathsf{t}, \mathsf{f}\}$ of the events from E, with probability $\prod_{e \text{ s.t.} \nu(e)=\mathsf{t}} \pi(e) \times \prod_{e \text{ s.t.} \nu(e)=\mathsf{f}} (1 - \pi(e))$. Evaluate* cie *and* fie *nodes by keeping only the child edges whose Boolean formula is true under ν. Evaluate* ind *nodes by choosing to keep or delete every child edge according to the probability indicated on its edge label. Evaluate* mux *nodes by removing all of their children edges, except one chosen according to its probability (possibly keep none if the probabilities sum up to less than 1). Finally, evaluate* det *nodes by replacing them by the collection of their children.*

All probabilistic choices are performed independently, so the overall probability of an outcome is the product of the probabilities at each step. Whenever an edge is removed, all of the descendant nodes and edges are removed. The probability of a document W according to D, written $D(W)$, is the total probability of all outcomes[4] *leading to W.*

Of course, the expressiveness and compactness of PrXML frameworks depend on which probabilistic nodes are allowed: we say that $\mathsf{PrXML}^{\mathcal{C}}$ is *more general* than $\mathsf{PrXML}^{\mathcal{D}}$ if there is a polynomial time algorithm to rewrite any $\mathsf{PrXML}^{\mathcal{D}}$ document to a $\mathsf{PrXML}^{\mathcal{C}}$ document representing the same probability distribution. Fig. 1 (adapted from [7]) represents this hierarchy on the PrXML classes that we consider.

[3] The non-standard constraint $x < 1$ means that ind does not subsume det (see Thm. 3 and 4).

[4] Note that in general there may be multiple outcomes that lead to the same document W.

Problem		Complexity	
Poss $\top$ fie	NP	(Prop. 1)
#Poss $\top$ fie	FP$^{\#P}$	(Prop. 1)
#Poss $<$ mux, ind, det	PTIME	(Thm. 1)
#Poss $\not<$ ind or mux	#P-hard	(Thm. 2)
Poss $\top$ ind or mux	PTIME	(Thm. 3)
Poss $\not<$ 2 of mux, ind, det	NP-hard	(Thm. 4)
#EPoss $\top$ mux, ind, det	PTIME	(Thm. 5)
EPoss $\bot$ cie	NP-hard	(Thm. 6)
Poss $\bot$ mie	NP-hard	(Thm. 7)
#EPoss $\top$ mie	PTIME	(Thm. 8)

Table 1: Summary of results

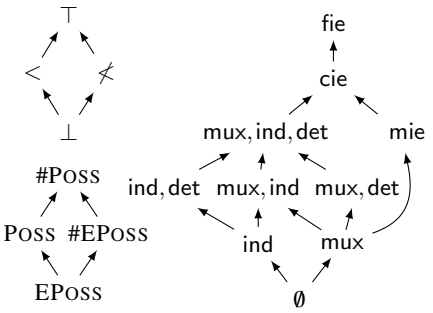

Fig. 1: Variants and PTIME reductions

3 Problem and general bounds

We now define the Poss problem formally, in its decision and computation variants.

Definition 4. *Given a class* PrXML$^{\mathcal{C}}$*, the* possibility problem *for* unordered *documents* Poss$^{\mathcal{C}}_{\not<}$ *is to determine, given as input an unordered* PrXML$^{\mathcal{C}}$ *document D and an unordered XML document W, whether W is a possible world of D, namely, $D(W) > 0$.*

The possibility problem *for* ordered *documents* Poss$^{\mathcal{C}}_{<}$ *is the same problem except that both D and W are ordered. For $o \in \{\not<, <\}$, the* #Poss$^{\mathcal{C}}_{o}$ *problem is the counting variant of* Poss$^{\mathcal{C}}_{o}$*, whose output is $D(W)$.*

For brevity, we write Poss$^{\mathcal{C}}_{\bot}$ and Poss$^{\mathcal{C}}_{\top}$ when describing lower or upper complexity bounds that apply to both Poss$^{\mathcal{C}}_{<}$ and Poss$^{\mathcal{C}}_{\not<}$.

We start by giving straightforward bounds on the most general problem variants:

Prop. 1. Poss$^{\mathsf{fie}}_{\top}$ *is in NP and* #Poss$^{\mathsf{fie}}_{\top}$ *is in FP$^{\#P}$.*

Prop. 2. Poss$^{\mathsf{cie}}_{\bot}$ *is NP-complete, even when D has height* 3*.*

Local models on ordered documents are known to be tractable using tree automata:

Theorem 1 ([6]). #Poss$^{\mathsf{mux,ind,det}}_{<}$ *can be solved in polynomial time.*

4 Local models

We now complete the picture for the local model PrXML$^{\mathsf{mux,ind,det}}$ on unordered documents. The results of [6] cannot be applied to this setting, as the ambiguity of node labels imply that we cannot impose an arbitrary order on document nodes; indeed, a reduction from perfect matching counting on bipartite graphs shows that the computation variant is hard even on the most inexpressive classes:

Theorem 2. #Poss$^{\mathsf{ind}}_{\not<}$ *and* #Poss$^{\mathsf{mux}}_{\not<}$ *are #P-hard, even when D has height* 4*.*

By contrast, the decision variant is tractable for PrXML$^{\mathsf{ind}}$ and PrXML$^{\mathsf{mux}}$, using a dynamic algorithm. However, allowing both ind and mux, or allowing det nodes, leads to intractability (by reductions from set cover and Boolean satisfiability).

Theorem 3. Poss$^{\mathsf{ind}}_{\top}$ *and* Poss$^{\mathsf{mux}}_{\top}$ *can be decided in PTIME.*

Theorem 4. Poss$^{\mathsf{ind,det}}_{\not<}$, Poss$^{\mathsf{mux,det}}_{\not<}$ *and* Poss$^{\mathsf{mux,ind}}_{\not<}$ *are NP-complete, even when D has height* 4*.*

5 Explicit matches

We now attempt to understand how the overall hardness of POSS is caused by the difficulty of finding how the possible world W can be *matched* to D.

Definition 5. *A* candidate match *of W in D is an injective mapping f from the nodes of W to the regular nodes of D such that, if r is the root of W then $f(r)$ is the root of D, and if n is a child of n' in W then there is a descending path from $f(n)$ to $f(n')$ going only through probabilistic nodes.*

Intuitively, candidate matches are possible ways to generate W from D, ignoring probabilistic annotations, assuming we can keep exactly the regular nodes of D that are in the image of f. There are exponentially many candidate matches in general, so it is natural to ask whether POSS is tractable if all matches are explicitly provided as input:

Definition 6. *Given a class* $\mathsf{PrXML}^{\mathcal{C}}$ *and $o \in \{\bot, \not<, <, \top\}$, the* POSS *problem with* explicit matches $\text{EPOSS}_o^{\mathcal{C}}$ *is the same as the* $\text{POSS}_o^{\mathcal{C}}$ *problem except that the set of the candidate matches of W in D is provided as input (in addition to D and W).*

The explicit matches variant generalizes many practical cases where all possible matches can be computed in polynomial time; for instance, when node labels are assumed to be unique, or *unambiguous* in the sense that no two sibling nodes carry the same label.

We first note that explicit matches ensure tractability of all local dependency models, by reduction to deterministic tree automata [6], this time also for unordered documents. Intuitively, we can consider all candidate matches separately and compute the probability of each one, in which case no label ambiguity remains so any order can be imposed:

Theorem 5. $\#\text{EPOSS}_{\top}^{\mathsf{mux,ind,det}}$ *can be solved in polynomial time.*

For long-distance dependencies, however, it is easily seen that POSS is still hard with conjunction of events, even if explicit matches are provided:

Theorem 6. $\text{EPOSS}_{\bot}^{\mathsf{cie}}$ *is NP-complete, even when D has height* 3*.*

This being said, it turns out that the hardness is really caused by event *conjunctions*. To see this, we introduce the $\mathsf{PrXML}^{\mathsf{mie}}$ class, which allows only *individual* events:

Definition 7. *The* $\mathsf{PrXML}^{\mathsf{mie}}$ *class features* multivalued independent events *taking their values from a finite set V (beyond* $\mathfrak{t}$ *and* $\mathfrak{f}$*, with probabilities summing to* 1*), and probabilistic* mie *nodes whose child edges are annotated by a* single *event e and a value $x \in V$. A* mie *node cannot be the child of a* mie *node. When evaluating D under a valuation ν, child edges of* mie *nodes labeled (e,x) should be kept if and only if $\nu(e) = x$.*

Note that mie hierarchies are forbidden (because they can straightforwardly encode conjunctions), so that $\mathsf{PrXML}^{\mathsf{mie}}$ does not capture ind hierarchies. However, as we introduced it with multivalued (not just Boolean) events, it captures $\mathsf{PrXML}^{\mathsf{mux}}$:

Prop. 3. *We can rewrite* $\mathsf{PrXML}^{\mathsf{mux}}$ *to* $\mathsf{PrXML}^{\mathsf{mie}}$ *and* $\mathsf{PrXML}^{\mathsf{mie}}$ *to* $\mathsf{PrXML}^{\mathsf{cie}}$ *in PTIME.*

In the $\mathsf{PrXML}^{\mathsf{mie}}$ class, the POSS problem is still NP-hard, by reduction to exact cover; however, with explicit matches, the #POSS problem is tractable, both in the ordered and unordered setting, despite the long-distance dependencies. Intuitively, the candidate matches are mutually exclusive, and each match's probability can be computed as that of a conjunction of equalities and inequalities on the events at the frontier.

Theorem 7. $\text{POSS}_{\bot}^{\mathsf{mie}}$ *is NP-complete, even when D has height* 3 *and events are Boolean.*

Theorem 8. $\#\text{EPOSS}_{\top}^{\mathsf{mie}}$ *can be solved in polynomial time.*

6 Conclusion

We have characterized the complexity of the counting and decision variants of POSS for unordered or ordered XML documents, and various PrXML classes. With explicit matches, #POSS is tractable unless event conjunctions are allowed. Without explicit matches, POSS is hard unless dependencies are local; in this case, if the documents are ordered, #POSS is tractable, otherwise #POSS is hard and POSS is tractable only with ind *or* mux nodes (and hard if both types, or det nodes, are allowed). Our results are summarized in Table 1 on page 4.

Further work could study more precisely the effect of det nodes and ind hierarchies, for instance by attempting to extend the $\mathsf{PrXML}^{\mathsf{mie}}$ class to capture them, or try to understand whether there is a connection between the algorithms of [6] and the proof of Thm. 3. It would also be interesting to determine under which conditions (beyond unique labels) can candidate matches be enumerated in polynomial time, so that the POSS problem reduces to the explicit matches variant. Last but not least, another natural problem setting is to allow the order on sibling nodes of D to be partly specified. This question is already covered in [6], but only when all of the possible orderings are explicitly enumerated: investigating the tractability of POSS for more compact representations, such as partial orders, is an intriguing problem.

Acknowledgements. The author thanks Pierre Senellart for careful proofreading, useful suggestions, and insightful feedback, the anonymous referees for their valuable comments, and M. Lamine Ba and Tang Ruiming for helpful early discussion. This work has been partly funded by the French government under the X-Data project.

References

1. A. Amarilli. The possibility problem for probabilistic XML (extended version). *CoRR*, 2014. `http://arxiv.org/abs/1404.3131`.
2. A. Amarilli and P. Senellart. On the connections between relational and XML probabilistic data models. In *Proc. BNCOD*, pages 121–134, Oxford, United Kingdom, 2013.
3. M. L. Ba, T. Abdessalem, and P. Senellart. Merging uncertain multi-version XML documents. *Proc. DChanges*, 2013.
4. M. L. Ba, T. Abdessalem, and P. Senellart. Uncertain version control in open collaborative editing of tree-structured documents. In *Proc. DocEng*, pages 27–36, 2013.
5. P. Barceló, L. Libkin, A. Poggi, and C. Sirangelo. XML with incomplete information. *JACM*, 58(1):4, 2010.
6. S. Cohen, B. Kimelfeld, and Y. Sagiv. Running tree automata on probabilistic XML. In *Proc. PODS*, pages 227–236. ACM, 2009.
7. E. Kharlamov, W. Nutt, and P. Senellart. Updating probabilistic XML. In *Proc. Updates in XML*, Lausanne, Switzerland, 2010.
8. B. Kimelfeld, Y. Kosharovsky, and Y. Sagiv. Query evaluation over probabilistic XML. *VLDB Journal*, 18(5):1117–1140, 2009.
9. B. Kimelfeld and P. Senellart. Probabilistic XML: Models and complexity. In Z. Ma and L. Yan, editors, *Advances in Probabilistic Databases for Uncertain Information Management*, pages 39–66. Springer-Verlag, 2013.
10. L. Libkin. *Elements of Finite Model Theory*. Springer, 2004.
11. D. Suciu, D. Olteanu, C. Ré, and C. Koch. *Probabilistic Databases*. Morgan & Claypool, 2011.