Ontology-mediated Queries over Probabilistic Data via Probabilistic Logic Programming

Timothy van Bremen
timothy.vanbremen@cs.kuleuven.be
KU Leuven, Belgium

Anton Dries
anton.dries@cs.kuleuven.be
KU Leuven, Belgium

Jean Christoph Jung
jeanjung@uni-bremen.de
Universität Bremen, Germany

ABSTRACT

We study ontology-mediated querying over probabilistic data for the case when the ontology is formulated in $\mathcal{ELH}^{dr}$, an expressive member of the $\mathcal{EL}$ family of description logics. We leverage techniques that have been developed (i) for classical ontology-mediated querying and (ii) for probabilistic logic programming and provide an implementation based on our findings. We include both theoretical considerations and an experimental evaluation of our approach.

ACM Reference Format:

1 INTRODUCTION

In many domains, data is inherently probabilistic due to uncertainty in the measurement or extraction process: for example, temperature readings from an unreliable sensor, or information extracted from the Internet using an imperfect NLP system. We nevertheless may be in possession of some deterministic domain knowledge which can be used to enrich the data. For instance, we may know that it cannot be snowing if we record the weather as being hot, or that every person extracted with the NLP system must have a date of birth. This idea of accessing uncertain data enriched with deterministic knowledge (an ontology) has led to the framework of ontology-mediated querying over probabilistic data (OMQPD) [12].

The main reasoning task in this setting is to evaluate a query over the data.

In this paper we focus on the ontology language $\mathcal{ELH}^{dr}$, an expressive member of the $\mathcal{EL}$ family of description logics [5], which forms the basis of the OWL 2 EL profile [1]. In contrast to $\mathcal{DL}$, first-order rewritings are not a complete method for ontology-mediated querying with ontologies formulated in $\mathcal{EL}$ (or extensions thereof). That is, there exist query/ontology pairs for which no equivalent first-order rewriting exists [7].

In order to have a complete tool, we show how the combined approach to ontology-mediated querying over deterministic data [13] can be lifted to the probabilistic case. This results in a rewriting of the query, ontology, and probabilistic data into a probabilistic logic program (PLP), which is a logic program augmented with uncertainty; see [9] for a recent survey. Often based on the distribution semantics, PLPs feature a combination of uncertainty and deterministic rules similar to the OMQPD setting. From a practical perspective, our rewriting technique allows us to take advantage of the extensive research conducted in the probabilistic logic programming community, which has resulted in several powerful inference systems, e.g., PRISM [15], cplint [2] and ProbLog [14]. As a proof-of-concept, we implemented the rewriting approach using ProbLog as an inference system and evaluated it on a probabilistic variant of the Lehigh University Benchmark [11]. From a theoretical perspective our approach also yields a polynomial time reduction from OMQPD to weighted model counting over propositional formulas, a popular approach for probabilistic inference [8].

2 BACKGROUND

We briefly review the description logic $\mathcal{ELH}^{dr}$. Fix disjoint countably infinite sets of concept and role names $\mathcal{C}_{C}$ and $\mathcal{R}_{R}$, respectively. Then $\mathcal{EL}$-concepts are formed according to the syntax rule

$$C ::= T | A \sqcap C | \exists r.C$$

where $A \in \mathcal{C}_{C}$ and $r \in \mathcal{R}_{R}$. An $\mathcal{ELH}^{dr}$-ontology (hereafter ontology) is a set of concept inclusions $C \subseteq D$, role inclusions $r \subseteq s$, domain restrictions dom($r$) $\subseteq C$, and range restrictions ran($r$) $\subseteq C$, where $C$ and $D$ are $\mathcal{EL}$-concepts and $r, s \in \mathcal{R}_{R}$. An ABox is a finite set of concept assertions $A(a)$ and role assertions $r(a, b)$ where $A \in \mathcal{C}_{C}$, $r \in \mathcal{R}_{R}$, and $a, b$ range over a countably infinite set of individual names $\mathcal{N}$. We denote with Ind($\mathcal{A}$) the set of all individual names that occur in $\mathcal{A}$. The semantics of $\mathcal{ELH}^{dr}$ is defined as usual in terms of interpretations $I = (\Delta^{I}, \cdot^{I})$; we elide a full description here and instead refer the reader to Baader et al. [6] for details. Following [5, 13], we assume without loss of generality that (i) for every role name $r$, there is exactly one range restriction $\text{ran}(r) \subseteq A$ in $T$, and we denote the concept name $A$ with $\text{ran}(r)$,

(ii) if $T \models r \subseteq s$, then $T \models \text{ran}(r) \subseteq \text{ran}(s)$,

(iii) there are no $r \neq s$ with $T \models s \subseteq r$ and $T \models r \subseteq s$,

(iv) for every domain restriction $\text{dom}(r) \subseteq C$, we have $C \in \mathcal{C}_{C}$, and (v) every $C \subseteq D$ in $T$ takes one of the following forms, for concept names $A, A', B$:

$$A \sqcap A' \subseteq B, \exists r.A \subseteq B, \ A \sqsubseteq \exists r.B.$$
Ontology-mediated Querying over Probabilistic Data. Let \( N_I \) denote a countably infinite set of variables disjoint from \( N_T \). Then \( N_T = N_T \cup N_I \) forms a set of terms. A conjunctive query (CQ) \( \phi \) is a first-order formula of the form \( \exists y. \psi(x, y) \), where \( x \) and \( y \) are tuples of variables in \( N_T \), and \( \psi(x, y) \) is a conjunction of atoms over signature \( N_C \cup N_K \) using terms from \( N_T \). We drop the free variables \( x \) of \( \phi(x) \) whenever no confusion can arise. An ontology-mediated query (OMQ) is a pair \((T, \phi)\) of ontology \( T \) and CQ \( \phi \). Given an ABox \( \mathcal{A} \), and an OMQ \((T, \phi)\), we say that a tuple of individuals from \( \mathcal{A} \) is a certain answer for \((T, \phi)\) over \( \mathcal{A} \) if \((T, \mathcal{A}) \models \phi(a)\), that is, every model \( I \) of \( T \) and \( \mathcal{A} \) satisfies \( I \models \phi(a) \). The set of all certain answers to \((T, \phi)\) is denoted by \( \text{cert}_\mathcal{A}(T, \phi) \).

Following [12], we use assertion-independent probabilistic ABoxes (ipABoxes) to model uncertain data. An ipABox is a pair \((\mathcal{A}, p)\) where \( \mathcal{A} \) is a classical ABox and \( p : \mathcal{A} \to [0, 1] \) assigns a probability to every assertion in \( \mathcal{A} \). An ipABox \((\mathcal{A}, p)\) induces a distribution \( p(\cdot) \) over possible ABoxes \( \mathcal{A}' \subseteq \mathcal{A} \), which is defined by taking

\[
p(\mathcal{A}') = \Pi_{a \in \mathcal{A}} p(a) \cdot \Pi_{a \in \mathcal{A} \setminus \mathcal{A}'} (1 - p(a)),
\]

for every \( \mathcal{A}' \subseteq \mathcal{A} \). The probability of an answer \( a \) to an OMQ \((T, \phi)\) over an ipABox \((\mathcal{A}, p)\) is then defined as:

\[
\text{Pr}_{\mathcal{A}, p}(T, \phi, a) = \sum_{\mathcal{A}' \subseteq \mathcal{A}, a \in \text{cert}_{\mathcal{A}'}(T, \phi)} p(\mathcal{A}').
\]

The prime inference task here is to compute answer probabilities, that is, given an ipABox \((\mathcal{A}, p)\) and an OMQ \((T, \phi)\) with answer candidate \( a \), compute \( \text{Pr}_{\mathcal{A}, p}(T, \phi, a) \).

Probabilistic Logic Programs. We introduce a variant of probabilistic logic programs that is sufficient for our purposes, though some systems support more features. A probabilistic logic program (PLP) is a triple \((\mathcal{T}, \varphi, \Pi)\) where \( \mathcal{T} \) is a set of facts, \( p : \mathcal{T} \to [0, 1] \) assigns a probability to every fact, and \( \Pi \) is a stratified logic program consisting of rules of the form:

\[
H \leftarrow B_1, \ldots, B_m, \neg B_{m+1}, \ldots, \neg B_n
\]

where \( H \) and all \( B_i \) are relational atoms over variables and individual names. The semantics of PLPs \((\mathcal{T}, \varphi, \Pi)\) is defined as follows. The pair \((\mathcal{T}, \varphi)\) induces a probability distribution \( p(\cdot) \) over subsets \( F' \subseteq F \) just as in Equation (1). Moreover, given a set of facts \( \mathcal{F} \) and a set of rules \( \Pi \), we denote with \( \Pi(\mathcal{F}) \) the minimal supported model of \( \mathcal{F} \cup \Pi \), obtained via the iterated fixed point construction of [3]. The prime inference task for PLPs is marginal inference, that is, compute the probability of a ground fact \( G \) under the PLP \((\mathcal{T}, \varphi, \Pi)\), which is:

\[
\text{Pr}_{\mathcal{T}, \varphi}(G) = \sum_{\mathcal{F} \subseteq \mathcal{T}, \text{G independent of } \mathcal{F}} p(\mathcal{F}).
\]

3 COMBINED APPROACH OVER IPABOXES

Let us briefly review the combined approach to OMQs over non-probabilistic ABoxes [13]. For simplicity, we use a slightly different presentation than in the original paper, but they are easily seen to be equivalent. In a nutshell, given \((T, \phi)\) and \( \mathcal{A} \), one computes in polynomial time an ABox \( \mathcal{A}_T \supseteq \mathcal{A} \) and a set of rules \( \Pi_{T, \phi} \) such that, for some distinguished relation name \( \text{Goal} \), we have

\[
\text{cert}(T, \phi) = \{ a \mid \text{Goal}(a) \in \Pi_{T, \phi}(\mathcal{A}_T) \}.
\]

Thus, the computation of certain answers is reduced to computing \( \mathcal{A}_T \) and the minimal model \( \Pi_{T, \phi}(\mathcal{A}_T) \). We will give more details on how \( \mathcal{A}_T \) and \( \Pi_{T, \phi} \) are constructed when they are needed to prove correctness of our adaptation. Note that the combined approach is not directly applicable to answering an OMQ \((T, \phi)\) over an ipABox \((\mathcal{A}, p)\) since for every \( \mathcal{A}' \subseteq \mathcal{A} \), the ABox \( \mathcal{A}' \) might be different. We solve this by providing a set of rules \( \Pi_{T, \phi} \) such that for every \( \mathcal{A}' \subseteq \mathcal{A} \), we have \( \Pi_{T, \phi}(\Pi_{T}(\mathcal{A}')) = \Pi_{T, \phi}(\mathcal{A}'') \).

Given (1), it is not hard to verify that we can use \( \Pi_{T, \phi} \) to lift the combined approach to answering OMQs over ipABoxes. More precisely, the following Lemma is an immediate consequence of the definition of marginal probabilities and (1):

**Lemma 1.** For every OMQ \((T, \phi)\), ipABox \((\mathcal{A}, p)\), and answer candidate \( a \), we have:

\[
\text{Pr}_{\mathcal{A}, p}(T, \phi, a) \cdot \text{Pr}_{\Pi_{T, \phi} \cup \Pi_{T}(\mathcal{A}')} \left( \text{Goal}(a) \right).
\]

In the construction of \( \Pi_T \), we use fresh individuals \( a_{A, B} \), where \( A, B \) are concept names that occur in \( T \), and a fresh concept name Aux to mark these individuals as auxiliary. Then, \( \Pi_T \) is the collection of the following rules (which do not use negation):

- \( a(a_{A,B}), B(a_{A,B}) \), and Aux\((a_{A,B})\), for every fresh \( a_{A,B} \);
- \( B(x) \leftarrow A(x), A'(x) \), for every \( A \cap A' \subseteq B \in T \);
- \( B(x) \leftarrow A(y), r(x, y) \), for every \( \exists r.a \subseteq B \in T \);
- \( r(x, a_{A,B}, \text{ran}(r)) \leftarrow A(x) \), for every \( A \subseteq \exists r.B \in T \);
- \( s(x, y) \leftarrow r(x, y) \), for every \( r \subseteq T \);
- \( A(x) \leftarrow r(x, y) \), for every \( \forall r(a) \subseteq A \in T \);
- \( A(x) \leftarrow r(x, y) \), for every \( \exists r(a) \subseteq A \in T \).

In order to prove (1), we recall the construction of \( \Pi_T \) and the relevant properties of the PLP \( \Pi_{T, \phi} \), as used in the combined approach. The extension \( \Pi_T \) of \( \mathcal{A} \) contains the following assertions:

- Aux\((a_{A,B})\), for every fresh \( a_{A,B} \);
- \( B(a) \), for all \( a \in \text{Ind}(\mathcal{A}) \) and \( B \) such that \( T \cup \mathcal{A} \models B(a) \);
- \( A'(a_{A,B}) \), for all \( a_{A,B} \) such that \( T \models A \cap B \subseteq A' \);
- \( r(a, b) \), for all \( a, b \in \text{Ind}(\mathcal{A}) \) with \( s(a, b) \in \mathcal{A} \) and \( T \models s \subseteq r \);
- \( r(a, a_{A,B}) \), for all \( a \in \text{Ind}(\mathcal{A}) \) such that \( T \cup \mathcal{A} \models \exists a.A(a), B = \text{ran}(s), \text{and } T \models s \subseteq r \);
- \( r(a_{A,B}, a_{A', B'}) \) for all \( A, B, A', B' \) such that \( T \models A \cap B \subseteq \exists s.A, B' = \text{ran}(s), \text{and } T \models s \subseteq r \).

It is routine to verify that \( \Pi_T(\mathcal{A}') = \mathcal{A}' \), for every \( \mathcal{A}' \subseteq \mathcal{A} \). Thus, in order to prove Property (1) it suffices to note that the rules in \( \Pi_{T, \phi} \) constructed in the classical combined approach do not use symbols from \( T, \mathcal{A}, \varphi \) in the head.

Lemma 2. For every OMQ \((T, \phi)\), ipABox \((\mathcal{A}, p)\), and possible answer \( a \), one can compute in polynomial time a propositional formula \( \chi \) and a weight function \( W \) such that \( \text{Pr}(\chi) = \text{Pr}_{\mathcal{A}, p}(T, \phi, a) \).
4 IMPLEMENTATION AND EXPERIMENTS

Lemma 1 immediately gives rise to an implementation of OMQPD with \( \mathcal{ELH}^{dr} \) on top of a probabilistic logic programming system supporting marginal inference. More precisely, on input \((\mathcal{T}, \varphi), (\mathcal{A}, \rho)\) and \(a\), we construct the PLP \( \mathcal{P} = (\Pi_{\mathcal{T}, \varphi} \cup \Pi_{\mathcal{T}, \varphi}, (\mathcal{A}, \rho)) \), feed the system with \( \mathcal{P} \), and query the marginal probability of \( \text{Goal}(a) \).

As a probabilistic logic programming system, we use \text{ProbLog} [14], which has a sufficiently expressive language with the required semantics. Moreover, it supports marginal inference via a variety of different algorithms. The overall architecture of our approach is depicted in Figure 1. For our experiments, we used two different inference methods: (i) the “classic” \text{ProbLog} inference approach of cycle-breaking and compilation to sentential decision diagrams (SDDs) [18], and (ii) \( T_{p} \)-compilation to SDDs, which avoids the cycle-breaking step altogether through forward inference [17]. Regardless of the method used, \text{ProbLog} first computes the ground program relevant to the query.

We conducted our experiments on a probabilistic version of the Lehigh University Benchmark (LUBM) [11]. LUBM is a benchmark for measuring the performance of semantic knowledge base systems in a consistent manner, comprising an ontology, data generation tool, and a set of test queries. For the purposes of our experiments, we dropped transitive and inverse role declarations from the ontology in order to obtain a valid \( \mathcal{ELH}^{dr} \)-ontology. We set the parameters of the original data generation tool to generate an ABox of cardinality 15189. Of this, 12260 statements were role assertions and the remainder were concept assertions.

We wrote scripts to transform the assertions generated by the data generation tool to probabilistic facts in \text{ProbLog}. As the data from the tool is deterministic by default, we enriched the output by associating each ABox statement with an independent, uniformly drawn probability \( X_{i} \sim U(0, 1) \) to obtain an ipABox. Finally, we computed the combined rewritings of each of the LUBM queries with respect to the ontology. As first-order rewritings exist for all of the queries we considered, we manually computed these rewritings for comparison purposes. Queries 11, 12, and 13 were deliberately omitted as they are specifically designed to test reasoning with inverse and transitive role declarations, which as mentioned earlier are unsupported in our ontology language. The results of applying \text{ProbLog} to the rewritten LUBM queries can be found in Table 1.

Interestingly, we see that, particularly in the combined approach, most of the time is spent in the grounding step rather than the knowledge compilation step for each query. These steps correspond to the (deterministic) query answering phase and probability computation phase, respectively. This means that a large amount of
time is taken in the computation of the relevant ground program, which is based on SLD-resolution. As SLD-resolution is, theoretically, not a hard task, we believe this to be the result of inefficiencies in ProbLog’s implementation of grounding which become apparent when dealing with large programs like the ones here.

The classic ProbLog inference method of cycle-breaking and compilation to SDDs consistently outperforms $T_P$-compilation. We also observe that first-order rewritings seem to have somewhat better inference times overall, as a trade-off for the incompleteness of this approach. We conclude that in practice, it may be best to first test the first-order rewritability of the query before resorting to the combined approach as a second option.

Finally, to get an indication of how our method scales, we examined the total inference time on different ABox sizes for a subset of the queries in Table 1 for which inference appeared non-trivial. The total inference time here is the sum of grounding, cycle-breaking, and SDD compilation time. The results are shown in Figure 2.

5 CONCLUSION AND FUTURE WORK

We established a connection between OMQPD with ontologies formulated in $\mathcal{ELH}^{\mathcal{TD}}$ and probabilistic logic programming, inspired by the combined approach known from classical ontology-mediated querying. We evaluated our approach with promising first results. There are a number of possible next steps. The results suggest that further work on ProbLog’s grounding engine is needed to scale to real-world database sizes; e.g. one could use the grounding constructed in Lemma 2. One could also investigate whether our approach extends to different ontology languages. Finally, it is interesting to see whether other inference capabilities of ProbLog, such as learning, can be transferred to the OMQPD setting.

ACKNOWLEDGMENTS

This work has received funding from the Research Foundation—Flanders (grant G042815N), and from the European Research Council under the European Union’s Horizon 2020 research and innovation programme (grant 694980).

REFERENCES


---

Table 1: Grounding and compilation runtime for the Lehigh University Benchmark queries. All times are in seconds.

<table>
<thead>
<tr>
<th>Query</th>
<th>Grounding</th>
<th>$T_P$-compilation</th>
<th>Cycle-breaking</th>
<th>Compilation</th>
<th>Grounding</th>
<th>$T_P$-compilation</th>
<th>Cycle-breaking</th>
<th>Compilation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.04</td>
<td>0.05</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>2</td>
<td>70.14</td>
<td>5.17</td>
<td>0.00</td>
<td>0.00</td>
<td>28.82</td>
<td>0.11</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>3</td>
<td>0.03</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.59</td>
<td>0.67</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>4</td>
<td>25.60</td>
<td>5.73</td>
<td>0.02</td>
<td>0.03</td>
<td>0.88</td>
<td>0.95</td>
<td>0.02</td>
<td>0.03</td>
</tr>
<tr>
<td>5</td>
<td>28.24</td>
<td>28.04</td>
<td>1.60</td>
<td>2.53</td>
<td>2.39</td>
<td>5.66</td>
<td>0.40</td>
<td>1.05</td>
</tr>
<tr>
<td>6</td>
<td>25.61</td>
<td>71.23</td>
<td>2.92</td>
<td>6.30</td>
<td>4.09</td>
<td>50.12</td>
<td>2.23</td>
<td>5.67</td>
</tr>
<tr>
<td>7</td>
<td>78.49</td>
<td>6.26</td>
<td>0.04</td>
<td>0.05</td>
<td>4.53</td>
<td>5.44</td>
<td>0.02</td>
<td>0.05</td>
</tr>
<tr>
<td>8</td>
<td>30.24</td>
<td>92.90</td>
<td>3.46</td>
<td>7.47</td>
<td>6.19</td>
<td>71.90</td>
<td>2.54</td>
<td>6.91</td>
</tr>
<tr>
<td>9</td>
<td>Timeout</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>Timeout</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>10</td>
<td>27.28</td>
<td>4.85</td>
<td>0.00</td>
<td>0.00</td>
<td>4.35</td>
<td>4.63</td>
<td>0.01</td>
<td>0.03</td>
</tr>
<tr>
<td>14</td>
<td>0.32</td>
<td>0.12</td>
<td>0.01</td>
<td>0.03</td>
<td>0.20</td>
<td>0.13</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

"Timeout" indicates that the procedure took over ten minutes to run.