Most Probable Explanations for Probabilistic Database Queries

İsmail İlkan Ceylan and Stefan Borgwardt
Faculty of Computer Science
Technische Universität Dresden, Germany
ismail.ceylan@tu-dresden.de
stefan.borgwardt@tu-dresden.de

Thomas Lukasiewicz
Department of Computer Science
University of Oxford, UK
thomas.lukasiewicz@cs.ox.ac.uk

Abstract

Forming the foundations of large-scale knowledge bases, probabilistic databases have been widely studied in the literature. In particular, probabilistic query evaluation has been investigated intensively as a central inference mechanism. However, despite its power, query evaluation alone cannot extract all the relevant information encompassed in large-scale knowledge bases. To exploit this potential, we study two inference tasks: namely finding the most probable database and the most probable hypothesis for a given query. As natural counterparts of most probable explanations (MPE) and maximum a posteriori hypotheses (MAP) in probabilistic graphical models, they can be used in a variety of applications that involve prediction or diagnosis tasks. We investigate these problems relative to a variety of query languages, ranging from conjunctive queries to ontology-mediated queries, and provide a detailed complexity analysis.

1 Introduction

Research on building large-scale probabilistic knowledge bases (PKBs) has resulted in a number of systems including NELL [Mitchell et al., 2015], Yago [Hoffart et al., 2013], DeepDive [Shin et al., 2015], and Google’s Knowledge Vault [Dong et al., 2014]. They have been used in a wide range of areas to automatically build structured knowledge bases. Most of them are based on the long tradition and foundations of probabilistic databases (PDBs) [Imieliski and Lipski, 1984; Suciu et al., 2011], which define probability distributions over sets of classical databases.

Probabilistic query evaluation is the key inference task underpinning these systems. However, PKBs encompass rich, structured knowledge, for which alternative inference mechanisms beyond probabilistic query evaluation are needed. Inspired by the maximal posterior probability computations in Probabilistic Graphical Models (PGMs) [Pearl, 1988; Koller and Friedman, 2009], we investigate the problem of finding most probable explanations for probabilistic queries to exploit the potential of such large databases to their full extent.

Computing the maximal posterior probability of a distinguished set of variables, given evidence about another set of variables, is one of the key computational problems in PGMs. In its general form, this problem is studied under the name of maximum a posteriori hypothesis\(^1\) (MAP), where the hypothesis refers to an instantiation of a set of distinguished variables. The most probable explanation (MPE) is a special case of MAP, where the distinguished variables and the evidence variables cover all variables in the model. Maximal posterior probability computations have also been lifted to relational probabilistic models such as Markov Logic Networks (MLNs) [Richardson and Domingos, 2006].

Both MPE and MAP translate to probabilistic databases in a natural way through the rich structure of queries. The most probable database problem (analogous to MPE), first proposed in [Gribkoff et al., 2014], is the problem of determining the (classical) database with the largest probability that satisfies a given query. Intuitively, the query defines constraints on the data, and the goal is to find the most probable database that satisfies these constraints. We also introduce a more intricate notion, called most probable hypothesis, which only asks for partial databases satisfying the query (analogous to MAP). The most probable hypothesis contains only tuples that contribute to the satisfaction condition of the query, which allows to more precisely pinpoint the most likely explanations of the query.

We study the computational complexity of the corresponding decision problems, denoted by MPD and MPH, respectively, for a variety of query languages. Our results provide detailed insights about the nature of these problems. We show that the data complexity of both problems is lower for existential queries than for universal queries. As expected, MPH usually has a higher complexity than MPD.

We extend our results towards ontology-mediated queries (OMQs), which enrich the well-known class of unions of conjunctive queries with the power of ontological rules based on Datalog\(^2\) (also called existential rules) [Baget et al., 2011; Cali et al., 2013]. This follows the tradition of ontology-based data access [Poggi et al., 2008], which allows us to query PDBs in a more advanced manner. For OMQs, our analysis shows that the computational complexity of these problems can change significantly w.r.t. the ontology languages

\(^1\)There exist other variants of these inference tasks [Kwisthout, 2011], and there are different naming conventions across communities; here, we use the terminology from the Bayesian Network (BN) literature [Darwiche, 2009].
under consideration and the complexity-theoretic assumptions employed. Our results provide tight complexity bounds for all Datalog languages known in the literature. Detailed proofs of all results can be found at \texttt{https://lat.inf.tu-dresden.de/research/papers.html}.

2 Background and Motivation

We recall the basics of classical query languages and databases and briefly describe the tuple-independent PDB model.

2.1 Queries and Databases

We consider a relational vocabulary consisting of finite sets \( R \) of predicates, \( C \) of constants, and \( V \) of variables. A term is a constant or a variable. An \textit{atom} is of the form \( P(s_1, \ldots, s_n) \), where \( P \) is an \( n \)-ary predicate, and \( s_1, \ldots, s_n \) are terms. A \textit{tuple} is an atom without variables.

A \textit{conjunctive query (CQ)} is an existentially quantified formula \( \exists x \phi \), where \( \phi \) is a conjunction of atoms, written as a comma-separated list. A \textit{union of conjunctive queries (UCQ)} is a disjunction of CQs. If \( \phi \) is an arbitrary Boolean combination of atoms, then we call \( \exists x \phi \) an existential query (\( \exists Q \)), and \( \forall x \phi \) a universal query (\( \forall Q \)). A \textit{first-order query (FOQ)} is an arbitrary first-order formula. A query is \textit{Boolean} if it has no free variables.

A \textit{database} is a finite set of tuples. The central problem studied for databases is \textit{query evaluation}: Finding all answers to a query \( Q \) over a database \( D \), which are assignments of the free variables in \( Q \) to constants such that the resulting first-order formula is satisfied in \( D \) in the usual sense. In the following, we consider only Boolean queries \( Q \), and focus on the associated decision problem, i.e., deciding whether \( Q \) is satisfied in \( D \), denoted as usual by \( D \models Q \).

2.2 Probabilistic Databases

The most elementary probabilistic database model is based on the tuple-independence assumption. We adopt this model and refer to \cite{Suciuetal} for details and alternatives. A probabilistic database induces a set of classical databases (called \textit{worlds}), each associated with a probability value.

Formally, a \textit{probabilistic database (PDB)} \( \mathcal{P} \) is a finite set of probabilistic tuples of the form \((t; p)\), where \( t \) is a tuple and \( p \in [0, 1] \), and, whenever \((t; p) \in \mathcal{P}\), \((t; q) \in \mathcal{P}\), then \( p = q \). A PDB assigns to every tuple \( t \) the probability \( p \) if \((t; p) \in \mathcal{P}\), and otherwise the probability 0. Since we usually consider a single, fixed PDB, we denote this probability assignment simply by \( \mathcal{P} \). We concentrate on the well-known possible worlds semantics. Under the tuple-independence assumption, \( \mathcal{P} \) induces the following \textit{unique probability distribution over classical databases} \( \mathcal{D} \):

\[
P(\mathcal{D}) := \prod_{t \in \mathcal{D}} P(t) \prod_{t \notin \mathcal{D}} (1 - P(t)).
\]

By the \textit{worlds induced by} \( \mathcal{P} \) we refer to those databases \( \mathcal{D} \) with \( P(\mathcal{D}) > 0 \). Query evaluation is also enriched by probabilistic information. More formally, the \textit{probability} of a Boolean query \( Q \) w.r.t. \( \mathcal{P} \) is \( P(Q) := \sum_{\mathcal{D} \models Q} P(\mathcal{D}) \).

**Example 1.** Consider the PDB \( \mathcal{P}_v \) given in Figure 1 and the conjunctive query

\[
Q_1 = \exists x, y \text{Veg}(x) \land \text{FriendOf}(x, y) \land \text{Veg}(y).
\]

Figure 1: The PDB \( \mathcal{P}_v \), where each table represents a predicate and each row is a probabilistic tuple.

through which we can ask the probability of vegetarians being friends with vegetarians. In the given PDB, alice, bob, and chris are all vegetarians and friends with each other (with some probability). The query

\[
Q'_1 = \exists y \text{Veg}(bob) \land \text{FriendOf}(bob, y) \land \text{Veg}(y)
\]

is a special case of \( Q_1 \) which asks whether bob has vegetarian friends. Its probability can be computed as \( P(Q'_1) = 0.9 \cdot 0.1 \cdot 0.6 = 0.054 \).

This task is known as \textit{probabilistic query evaluation}, and it has been studied extensively in PDBs. In a celebrated result by \cite{DalviSuciu}, it has been shown that UCQs exhibit a data complexity dichotomy between P and #P for probabilistic query evaluation. However, while dealing with large-scale knowledge bases, alternative inference mechanisms are needed. Consider again our running example, and suppose that we have observed that \( Q'_1 \) is true and would like to learn what best explains this observation w.r.t. the PDB \( \mathcal{P}_v \).

We revisit this example after providing a principled approach for dealing with such tasks.

3 Most Probable Explanations

We investigate the problem of finding \textit{most probable explanations} for probabilistic database queries under two different semantics. Finding the \textit{most probable database} is to determine the world with the largest probability that satisfies a given query, as formalized by \cite{Gribkoffetal}:

**Definition 2.** The \textit{most probable database} for a query \( Q \) over a PDB \( \mathcal{P} \) is

\[
\arg \max_{\mathcal{D} \models Q} P(\mathcal{D}),
\]

where \( \mathcal{D} \) ranges over all worlds induced by \( \mathcal{P} \).

Intuitively, a PDB defines a probability distribution over exponentially many classical databases, and the most probable database is the element in this collection that has the highest probability while still satisfying the query. This can be seen as the best instantiation of a probabilistic model, and hence analogous to MPE in BNs \cite{Darwiche}.\footnote{\textcopyright 2017 by The International Joint Conference on Artificial Intelligence (IJCAI-17)}

**Example 3.** Consider again the PDB \( \mathcal{P}_v \) and the query

\[
Q_2 = \forall x, y \neg \text{Veg}(x) \lor \neg \text{Eats}(x, y) \lor \neg \text{Meat}(y),
\]

which defines the constraints of being a vegetarian, which is violated by the tuples Veg(chris), Eats(chris, mussels) and Meat(mussels). Hence, the most probable database
for Q₂ cannot contain all three of them. It is easy to see that Veg[chris] is removed, as it has the lowest probability. Thus, the most probable database (in this case unique) contains all tuples of P, that have a probability above 0.5, except for Veg[chris]. Suppose that we have observed Q₁, and we are interested in finding an explanation for this observation under the constraint of Q₂, which is specified by Q₃ = Q₁ ∩ Q₂. In this case, the most probable database must contain the tuples Veg[chris] and Friend[chris].bob, whereas Eats[chris, mussels] is omitted. ■

Finding the most probable database is important, as it identifies the most likely state of a PDB relative to a query. However, it has certain limitations, which are analogous to the limitations of MPE in PGMs [Koller and Friedman, 2009]. Most importantly, one is always forced to choose a complete database although the query usually affects only a subset of the tuples. In other words, it is usually not the case that the whole database is responsible for the goal query to be satisfied. To be able to more precisely pinpoint the explanations of a query, we introduce the following more intricate notion.

**Definition 4.** The most probable hypothesis for a query Q over a PDB P is

\[
\text{MPH}(Q) = \arg \max_{H \in Q} \sum_{D \models H} P(D),
\]

where H ranges over sets of tuples t and negated tuples ¬t such that t occurs in P, and ⊨ denotes the open-world entailment relation, i.e., H ⊨ Q holds iff all worlds induced by P that satisfy H also satisfy Q.

The sum inside the maximization evaluates to the product of the probabilities of the (negated) tuples in H, and hence we denote it by P(H). Note that each D ⊨ H must satisfy Q, and thus the most probable database is a special case of the most probable hypothesis that has to contain all tuples from P. In contrast, the most probable hypothesis contains tuples only if they contribute to the satisfaction of the query.

**Example 5.** The most probable hypothesis H for Q₃ consists of Veg[bob], Veg[chris], Friend[chris].bob, and ¬Eats[chris, mussels], which yields a probability of P(H) = 0.9 · 0.6 · 0.1 · (1 − 0.8) = 0.0108. Since the most probable hypothesis contains less tuples, it is more informative than the most probable database for Q₃. ■

Whereas the most probable database represents full knowledge about all facts, which corresponds to the common closed-world assumption for (probabilistic) databases, the most probable hypothesis may leave tuples of P unresolved, which can be seen as a kind of open-world assumption (although the tuples that do not occur in P are still false).

**Complexity Results**

We formulate the decision problems MPD and MPH and investigate their computational complexity (see Table 1). We assume familiarity with computational complexity theory, and the distinction between data and combined complexity.

**Definition 6.** Let Q be a query, P a PDB, and p ∈ (0, 1] a threshold. We denote by MPD (resp., MPH) the problem of deciding whether there exists a database D (resp., hypothesis H) that satisfies Q with P(D) ≥ p (resp., P(H) ≥ p).

Our first result concerns the well-known class of UCQs, and we observe that both MPD and MPH remain in polynomial time in the data complexity and in NP for the combined complexity.

**Theorem 7.** MPD and MPH for UCQs can be decided in polynomial time in the data complexity and are NP-complete in the combined complexity.

Intuitively, polynomial-time data complexity is ensured by the fact that UCQs are monotone queries: as a simple algorithm for MPD, consider all matches (of which there are polynomially many) of a given UCQ, and extend each match with all (negated) tuples from the PDB that have a probability greater than 0.5. This results in (polynomially many) classical databases, one of which is the most probable database. The polynomial data complexity result for MPH follows from similar arguments.

This result may seem immediate, as UCQs are monotone queries, but for MPH we can show an even stronger result that applies to all existential queries, even if negations in front of query atoms are allowed. For MPD, we show an additional hardness in the combined complexity.

**Theorem 8.** MPD for ∃Q can be decided in polynomial time in the data complexity and is NP-complete in the combined complexity. MPH for ∃Q can be decided in \(\Sigma^p_2\) in the data complexity and is \(\Sigma^p_3\)-complete in the combined complexity.

Determining the precise data complexity of MPH for ∃Q is left as an open problem.

For universally quantified queries, MPD becomes harder even in the data complexity. Given the rich structure of the query, it becomes possible to encode non-deterministic choices over all possible databases. It is important to note that a similar hardness result was also obtained in [Gribkoff et al., 2014] (although on a different query).

**Theorem 9.** MPD for ∀Q is \(\Sigma^p_2\)-complete in the data complexity.

We observe a similar phenomenon for MPH, where the increase in the complexity is even more dramatic.

**Theorem 10.** MPH for ∀Q is \(\Sigma^p_3\)-complete in the data complexity.

Note that also MPE and MAP differ in terms of computational complexity: in BNs, the former is NP-complete [Shimony, 1994; Littman, 1999] and the latter NP\(^{PP}\)-complete [Park and Darwiche, 2004a]. Differently, MPH remains in \(\Sigma^p_2 \subseteq \text{NP}^{PP}\), since computing the probability of the hypothesis can be done in polynomial time due to the tuple-independence assumption. As we will examine later, allowing ontological rules to induce correlations on the tuples results in different complexities, which are more comparable to PGMs, which can express conditional dependencies between their variables. Our final result for this section concerns the combined complexity of both problems for ∀Q.

**Theorem 11.** MPD and MPH for ∀Q are \(\Sigma^p_2\)-complete in the combined complexity.

If we consider arbitrary FOQs, it is easy to show that the data complexity remains the same as for ∀Qs, and the combined complexity is still \(\text{PS} \text{SPACE}\), as for classical databases.
Moreover, all our combined complexity results for PDBs hold even in the case where the arity of the predicates is bounded by some constant, commonly known as the bounded-arity (ba) combined complexity (see Table 1). The reason is that our hardness results use only predicates of a bounded arity.

4 Most Probable Explanations for OMQs

We now study the problems in the presence of ontological rules, which add deductive power to PDBs. The benefits of using of ontologies for large-scale PKB completion are well known [Jung and Lutz, 2012; Borgwardt et al., 2017]. From a broader perspective, extending tuple-independent PDBs with logical rules is an old idea, aiming to induce correlations on a logical level, and thus to relax the tuple independence, resulting in very powerful formalisms [Poole, 1993; 1997].

In the remainder of this paper, we restrict ourselves to UCQs, but in exchange consider additional knowledge encoded through an ontology. In the simplest case, we can formulate negative constraints (NCs) like

\[ \forall x, y \text{ Veg}(x) \land \text{Eat}(x, y) \land \text{Meat}(y) \rightarrow \bot, \]

which imposes the same constraints as \( Q_{2} \). NCs are a special case of denial constraints over databases [Staworko and Chomicki, 2010]. We also allow to formulate more general ontological knowledge in the form of tuple-generating dependencies (TGDs). For instance, the first-order formulas

\[ \forall x, y \text{ FriendOf}(x, y) \rightarrow \text{FriendOf}(y, x), \]

\[ \forall x \text{ Veg}(x) \rightarrow \exists y \text{ Knows}(x, y) \land \text{Veg}(y) \]

are TGDs stating that the friend relation is symmetric and that every vegetarian knows another vegetarian. In particular, TGDs can express the well-known inclusion dependencies and join dependencies from database theory. Taking all parts together, we are talking about ontology-mediated queries, which are UCQs in combination with a so-called Datalog\(^{\pm}\) ontology, i.e., a finite set of NCs and TGDs [Calì et al., 2012]. We will pay particular attention to the case where only NCs are allowed, as this is closest to the constraints over PDBs that we described in the previous examples, and it gives us a baseline for the computational complexity.

More formally, an NC \( \nu \) is an FO formula \( \forall x \varphi(x) \rightarrow \bot, \) where \( \varphi(x) \) is a conjunction of atoms, called the body of \( \nu, \) and \( \bot \) is the truth constant \( \text{false}. \) A TGD \( \sigma \) is an FO formula \( \forall x \varphi(x) \rightarrow \exists y P(x, y), \) where \( \varphi(x) \) is a conjunction of atoms, called the body of \( \sigma, \) and \( P(x, y) \) is an atom, called the head of \( \sigma. \) A (Datalog\(^{\pm}\)) program (or ontology) \( \Sigma \) is a finite set of NCs and TGDs.\(^{2}\) An ontology-mediated query (OMQ) \( (Q, \Sigma) \) consists of a program \( \Sigma \) and a Boolean UCQ \( Q. \)

For the semantics, we extend the vocabulary by an infinite set \( N \) of nulls. An instance \( I \) is a possibly infinite set of tuples that may additionally contain nulls; it satisfies a TGD or NC \( \sigma \) if \( I \models \sigma, \) where \( \models \) denotes standard FO entailment. \( I \) satisfies a program \( \Sigma, \) written \( I \models \Sigma, \) if \( I \) satisfies each formula in \( \Sigma. \) The set \( \text{mods}(D, \Sigma) \) of models of a database \( D \) relative to a program \( \Sigma \) is \( \{ I \mid I \models D \text{ and } I \models \Sigma \}. \) A database \( D \) is consistent w.r.t. \( \Sigma \) if \( \text{mods}(D, \Sigma) \) is non-empty. The OMQ \( (Q, \Sigma) \)

is entailed by \( D, \) denoted \( D \models (Q, \Sigma), \) if \( I \models Q \) holds for all \( I \in \text{mods}(D, \Sigma). \) Observe that consistency of \( D \) w.r.t. \( \Sigma \) can thus be written as \( D \models (Q, \Sigma) \), or equivalently, \( D \models (Q_{1}, \Sigma^*) \), where \( Q_{1} \) is the UCQ obtained from the disjunction of the bodies of all NCs in \( \Sigma \), and \( \Sigma^* \) contains all TGDs of \( \Sigma. \) The probability of an OMQ over a PDB is defined as usual, by summing up over the consistent worlds that entail the query.

In general, the entailment problem is undecidable [Beeri and Vardi, 1981], which motivated syntactic fragments of TGDs (but not the NCs) [Calì et al., 2012; Baget et al., 2011; Krötzsch and Rudolph, 2011; Calì et al., 2013; Fugin et al., 2005]. We focus on only a few of these classes here (see Table 1). If there are no TGDs at all, i.e., \( \Sigma^* = \emptyset, \) then we denote the query language by OMQ\(_{NC}\). One of the most important classes, guarded programs (OMQ\(_{G}\)), allow only such TGDs that have a body atom (the guard) that contains all body variables. In the special case of linear programs (OMQ\(_{L}\)), the body may consist of only one atom. In frontier-guarded programs (OMQ\(_{FG}\)), only the frontier variables, i.e., those shared by the body and the head, need to be guarded. We pay particular attention to the class of guarded and full programs (OMQ\(_{GF}\)), which does not allow existentially quantified variables, and is one of the least expressive Datalog\(^{\pm}\)-based query languages with a \( \Pi_{2}\)-complete data complexity. At the upper end of the expressivity spectrum, we observe classes like weakly guarded (OMQ\(_{WG}\)) programs, whose data complexity is already \( \Pi_{2}\)-complete. From a theoretical perspective, the class of acyclic programs (OMQ\(_{A}\)), which do not allow cyclic dependencies between predicates, is of interest, because it has a non-deterministic combined complexity, which leads to a different behavior.

A key paradigm in OMQ answering is the FO-rewritability of queries: an OMQ \( (Q, \Sigma) \) is FO-rewritable if there exists a Boolean UCQ \( Q_{\Sigma} \) such that, for all consistent databases \( D, \) it holds that \( D \models (Q, \Sigma) \) iff \( D \models Q_{\Sigma}. \) Intuitively, the rewritten query \( Q_{\Sigma} \) can be answered directly over the database, without referring to the Datalog\(^{\pm}\) program. A class OMQ is FO-rewritable if all its OMQs are FO-rewritable; in this case, the OMQ entailment problem for OMQ\(_{G}\) has a data complexity of \( AC^{0}. \) Of the classes mentioned above, only OMQ\(_{NC}, \) OMQ\(_{A}, \) and OMQ\(_{A}\) have this property.

In addition to data complexity, bo-combined complexity, and combined complexity, for OMQs we also consider the fixed-program (fp) combined complexity, where the Datalog\(^{\pm}\) program is viewed as fixed, but both (P)DB and query are considered to be part of the input. The bo-combined complexity is of interest for Description Logics [Baader et al., 2003], some of which can be considered to be Datalog\(^{\pm}\) languages with arity at most 2.

4.1 Most Probable Databases for OMQs

For ontology-mediated queries, models are restricted to those that are consistent with the ontology. In other words, the constraints are considered separately from the (existential) query; thus, the definitions of MPD and MPH have to be adapted accordingly. The main difference is that we now consider only the consistent worlds induced by the PDB, and thus maximize only over consistent worlds.
Definition 12. The most probable database for an OMQ $(Q, \Sigma)$ over a PDB $P$ is

$$\arg \max_{D \models (Q, \Sigma), \text{mols}(D, \Sigma) \in \emptyset} P(D).$$

As before, we consider the decision problem MPD, which checks for the existence of a world $D$ as above that exceeds a given probability threshold $p$. Our complexity results for this problem are summarized in the lower left part of Table 1.

A naive approach to solve MPD is to guess the database $D$, and then check that it entails the given OMQ, does not entail the query $(Q_1, \Sigma^*)$, and exceeds the probability threshold. Since the probability can be computed in polynomial time, the problem can be decided by a single non-deterministic Turing machine using an oracle to check OMQ entailment.

Theorem 13. If entailment for OMQ$_X$ is in a complexity class $C$, then MPD for OMQ$_X$ is in $NP^C$ (under the same complexity assumptions).

By a reduction from 3-colorability, we show that MPD is NP-hard already in the data complexity, even if we only use NCs, i.e., the query and the positive program $\Sigma^*$ are empty. This strengthens our previous result about $\forall Q$, since NCs can be expressed by universal queries, but are not allowed to use negated atoms.

Theorem 14. MPD for OMQ$_{NC}$ is in $NP$-hard in the data complexity, even for $Q = \top$.

We show a matching upper bound even in the fp-combined complexity, by rethinking from Theorem 13. Since here the query $(Q_1, \Sigma^*)$ is fixed, for the non-entailment check, we can refer to the data complexity of OMQ entailment. Assuming that this is possible in P, and further that the fp-combined complexity does not exceed NP, then the whole test can be done by a single non-deterministic Turing machine in polynomial time. This insight yields an upper bound of NP for most of the classes we consider.

Theorem 15. If entailment for OMQ$_X$ is in $NP$ in the fp-combined complexity and in $P$ in the data complexity, then MPD for OMQ$_X$ is in $NP$ in the fp-combined complexity.

Under ba-combined complexity assumptions, we observe an increase in complexity, which intuitively comes from the fact that the query $(Q_1, \Sigma^*)$ is not fixed anymore.

Theorem 16. MPD for OMQ$_{NC}$ is $\Sigma_2^p$-hard in ba-combined complexity, even for $Q = \top$.

We obtain a matching upper bound from Theorem 13 if $C = NP$. Most of the remaining hardness results follow from the complexity of classical OMQ entailment, since an OMQ $(Q, \Sigma)$ is entailed by a consistent database $D$ iff the most probable database w.r.t. $\{t : 1 \mid t \in D\}$ has probability 1. In combination with Theorem 13, this yields tight complexity results for large deterministic classes like $PSPACE$ or $EXP$. This leaves open only the case of OMQ$_A$, for which entailment is $NEXP$-complete in the (ba-)combined complexity, which yields an upper bound of $NP^{NEXP} = P^{NEXP} = P^{NP}$ due to Theorem 13 and results from [Hemachandra, 1989]. We show that this bound is tight, using a reduction from a $P^{NEXP}$-complete version of the tiling problem [Fürer, 1983].

Theorem 17. MPH for OMQ$_A$ is $P^{NP}$-hard in the ba-combined complexity.

4.2 Most Probable Hypotheses for OMQs

Similarly to MPD, we have to update the definition of MPH to take into account only consistent worlds.

Definition 18. The most probable hypothesis for an OMQ $(Q, \Sigma)$ over a PDB $P$ is

$$\arg \max_{H \models (Q, \Sigma), \text{mols}(D, \Sigma) \in \emptyset} \sum_{D \models H} P(D),$$

where $H$ is a set of (non-probabilistic) tuples $t$ occurring in $P$.

Since we consider only monotone queries (i.e., UCQs), we do not have to include negated tuples in the hypothesis.

To solve MPH, one can guess a hypothesis, and then check whether it entails the query, and whether the probability mass of its consistent extensions exceeds the given threshold. The latter part can be done by a PP Turing machine with an oracle.

<table>
<thead>
<tr>
<th>Query Languages</th>
<th>Most Probable Database</th>
<th>Most Probable Hypothesis</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>data</td>
<td>$fp$-comb.</td>
</tr>
<tr>
<td>UCQ</td>
<td>in $P$</td>
<td>$NP$</td>
</tr>
<tr>
<td>$\exists Q$</td>
<td>in $P$</td>
<td>$NP$</td>
</tr>
<tr>
<td>$\forall Q$</td>
<td>$NP$</td>
<td>$\Sigma_2^p$</td>
</tr>
<tr>
<td>FOQ</td>
<td>$NP$</td>
<td>$PSpace$</td>
</tr>
<tr>
<td>OMQ$_{NC}$</td>
<td>$NP$</td>
<td>$NP$</td>
</tr>
<tr>
<td>OMQ$_{A}$</td>
<td>$NP$</td>
<td>$NP$</td>
</tr>
<tr>
<td>OMQ$_{GF}$</td>
<td>$NP$</td>
<td>$NP$</td>
</tr>
<tr>
<td>OMQ$_{FG}$</td>
<td>$NP$</td>
<td>$NP$</td>
</tr>
<tr>
<td>OMQ$_{WG}$</td>
<td>$Exp$</td>
<td>$Exp$</td>
</tr>
</tbody>
</table>

Table 1: Complexity results for MPD and MPH for a wide range of queries: all listed results are original contributions of this paper except the data complexity for $\forall Q$. This latter result has been strengthened towards the weaker representation OMQ$_{NC}$.
for OMQ entailment (by normalizing the probability of the worlds such that the threshold is exactly 0.5). The oracle can be used also for the initial entailment check.

**Theorem 19.** If entailment for $OMQ_X$ in a complexity class $C$, then MPH for $OMQ_X$ is in $NP^{PP^C}$ (under the same complexity assumptions).

For any $C \in PH$, this yields $NP^{PP^C} = NP^{PP} = NP^{PP}$ as an upper bound, due to a result from [Toda, 1989]. Except for $PP$, all other upper bounds in the lower right part of Table 1 also follow from this observation. For $C = NEXP$, the whole $PP^{NP^{EXP}}$ computation can be done by an NEXP oracle, and hence we again obtain $NP^{NP^{EXP}} = P^{NEXP}$ in this case.

On the other hand, we can transfer most of the lower bounds from MPD by a simple reduction.

**Theorem 20.** If MPD for $OMQ_X$ is hard for a complexity class $C$, then MPH for $OMQ_X$ is also hard for $C$ (under any complexity measure except for the data complexity).

We now discuss the more interesting cases below $PSPACE$. For FO-rewritable classes of OMQs, we obtain a data complexity of only $PP$. This is due to the fact that TGDs can be compiled into the query, and the observation of Theorem 8 that existential queries can be processed using polynomially many steps. Nevertheless, in each step, we have to compute a non-trivial sum over the consistent worlds. Consequently, we have to compute a non-trivial sum over the consistent worlds. Finally, the polynomially many $PP$ checks can be compiled into a single $PP$ operation [Beigel et al., 1995].

**Theorem 21.** If entailment for $OMQ_X$ is in $AC^0$ in the data complexity, then MPH for $OMQ_X$ is $PP$-complete in the data complexity.

To frame this result, we show $NP^{PP}$-hardness both for OMQ$_{GF}$ (the prototypical non-FO-rewritable class) in the data complexity, and for OMQ$_{NC}$ in the full-combined complexity. We use two similar reductions from a problem in the polynomial-time counting hierarchy [Wagner, 1986]. In the first one, we rely on the power of guarded and full TGDs; in the second, we use a non-fixed query and a careful choice of probabilities to simulate the effect of these TGDs.

**Theorem 22.** MPH is $NP^{PP}$-hard for OMQ$_{GF}$ in the data complexity, and for OMQ$_{NC}$ in the full-combined complexity.

We also observe a dichotomy in the data complexity of MPH, which follows from the dichotomy for probabilistic query evaluation [Dalvi and Suciu, 2012]. Note that the PP-hardness holds only under Turing reductions.

**Lemma 23** (Dichotomy). For FO-rewritable classes $OMQ_X$, MPH is either in $P$ or PP-hard in the data complexity.

Our results apply to all decidable Datalog$^+$ languages from the literature, due to the generic nature of our theorems and the known complexity results for OMQ entailment [Cali et al., 2012; Baget et al., 2011; Krötzsch and Rudolph, 2011; Cali et al., 2013]. For example, full sets of TGDs (F) behave like OMQ$_{GF}$, and linear full (LF) and acyclic full (AF) sets of TGDs exhibit the same complexities as OMQ$_L$.

5 Related Work

Our work is inspired by the maximal posterior probability computations in PGMs [Pearl, 1988; Koller and Friedman, 2009]. It is well-known that they are a form of abduction, which clearly also applies to the problems studied here. Maximal posterior probability computations are central in PGMs and in statistical relational learning. However, analogous problems have not been studied in depth in the context of PDBs. To our knowledge, the only work in this direction is on most probable databases [Gribkoff et al., 2014], which we use as a starting point.

Probabilistic query evaluation has been investigated in depth in the literature [Imieliski and Lipski, 1984; Suciu et al., 2011]. In fact, it is known to be either in $P$ or #P-hard for UCQs in the data complexity [Dalvi and Suciu, 2012]. Extensions of PDBs with ontological rules, in particular with Datalog$^+$, are also well-covered in the literature [Toda, 2013; Borgwardt et al., 2017]. The focus of these works is on probabilistic query answering.

Most of our results are covered by the classes $NP$, $\Sigma_2^P$, $PP$, and $NP^{PP}$. Though intractable, they are at the core of many important problems, which motivated a body of work tailored towards scalable algorithms for these classes. There is an immediate connection between MPE and weighted MAX-SAT [Sang et al., 2007]. Similarly, advances in knowledge compilation [Park and Darwiche, 2004b; Pipatsrisawat and Darwiche, 2009] and approximate model counting [Chakraborty et al., 2016; Fremont et al., 2017] are tailored to achieve optimal, scalable algorithms for problems in classes such as $PP$ and $NP^{PP}$. Both MPD and MPH can be cast into their propositional variants using the lineage representation of queries, which gives immediate access to such algorithms. However, there is need for future work in this direction, as grounding is not an optimal way of handling these problems; performing inference directly on FO-structures is known to be more efficient.

6 Summary and Outlook

We studied two inference tasks for PDBs; namely finding the most probable database and the most probable hypothesis for a given query. The focus of this paper was to determine the precise complexity of these problems, and we provided a detailed analysis for these problems relative to a variety of query languages, ranging from conjunctive queries to ontology-mediated queries. The main focus of future work is on the one hand to extend these results to other classes (such as equality generating dependencies) and on the other hand to obtain even more refined results (such as classification results).

Acknowledgments

This work was supported by DFG in the CRC 912 (HAEC), GRK 1907 (RoSI), and the project BA 1122/19-1 (GoAsQ), and by the UK EPSRC grants EP/J008346/1, EP/L012138/1, EP/M025268/1, and by The Alan Turing Institute under the EPSRC grant EP/N510129/1.

References
