Technische Universität Dresden
Institute for Theoretical Computer Science
Chair for Automata Theory

LTCS-Report

Ontology-Mediated Queries for Probabilistic Databases (Extended Version)

Stefan Borgwardt and İsmail İlkan Ceylan and Thomas Lukasiewicz

LTCS-Report 17-10

This is an extended version of the article in: Proc. of the 31st AAAI Conf. on Artificial Intelligence (AAAI'17), pages 1063–1069, AAAI Press, 2017.

Postal Address:
Lehrstuhl für Automatentheorie
Institut für Theoretische Informatik
TU Dresden
01062 Dresden

http://lat.inf.tu-dresden.de

Visiting Address:
Nöthnitzer Str. 46
Dresden

Ontology-Mediated Queries for Probabilistic Databases

Stefan Borgwardt and **İsmail İlkan Ceylan**
Faculty of Computer Science
Technische Universität Dresden, Germany
firstname.lastname@tu-dresden.de

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

Abstract

Probabilistic databases (PDBs) are usually incomplete, e.g., contain only the facts that have been extracted from the Web with high confidence. However, missing facts are often treated as being false, which leads to unintuitive results when querying PDBs. Recently, open-world probabilistic databases (OpenPDBs) were proposed to address this issue by allowing probabilities of unknown facts to take any value from a fixed probability interval. In this paper, we extend OpenPDBs by Datalog$^\pm$ ontologies, under which both upper and lower probabilities of queries become even more informative, enabling us to distinguish queries that were indistinguishable before. We show that the dichotomy between P and PP in (Open)PDBs can be lifted to the case of first-order rewritable positive programs (without negative constraints); and that the problem can become $\mathrm{NP}^{\mathrm{PP}}$-complete, once negative constraints are allowed. We also propose an approximating semantics that circumvents the increase in complexity caused by negative constraints.

1 Introduction

The effort for building *large-scale knowledge bases* from data in an automated manner has resulted in a number of systems including NELL (Mitchell et al. 2015), Yago (Hoffart et al. 2013), DeepDive (Shin et al. 2015), Microsoft's Probase (Wu et al. 2012), and Google's Knowledge Vault (Dong et al. 2014). They combine methods from information extraction, natural language processing, relational learning, and databases to process large volumes of uncertain data. The state of the art to store and process such data is founded on *probabilistic databases* (PDBs) (Imieliski and Lipski 1984; Fuhr and Rölleke 1997; Suciu et al. 2011).

Each of the above systems encodes only a portion of the real world, and this description is necessarily *incomplete*. Thus, a meaningful querying semantics must provide a way to deal with missing information. Recently, an effort in this direction was made by introducing *open-world probabilistic databases* (OpenPDBs) (Ceylan, Darwiche, and Van den Broeck 2016), which generalize PDBs to be able to deal with incompleteness. More precisely, in OpenPDBs the probabilities of facts that are not in the database, called *open tuples*, are relaxed to a default probability interval, which is very different from the *closed-world assumption* of PDBs, which requires the probabilities of such facts to be zero. In the resulting framework of OpenPDBs, query probabilities are given in terms of *upper* and *lower* probability values, which is more in line with an incomplete view of the world.

While forming a natural and flexible basis for querying incomplete data sources, OpenPDBs are limited in the following sense: All open tuples can take on probability values from a single *fixed* interval $[0, \lambda]$, which results in the *same* upper and lower probabilities for many queries. Consider, for instance, the PDB containing the probabilistic tuples $\langle \mathtt{Author}(a) : 0.8\rangle$, $\langle \mathtt{Pub}(a, b) : 0.6\rangle$, $\langle \mathtt{Pub}(c, d) : 0.9\rangle$, $\langle \mathtt{Novel}(d) : 1\rangle$. In OpenPDBs, $\mathtt{Author}(c)$ and $\mathtt{Author}(d)$ evaluate to the *same lower and upper probabilities* (0 and λ, respectively), since both tuples are open. Intuition, however, tells us that c is more likely to be an author, as we already know (with high confidence) that c has published a novel. On the other hand, $\mathtt{Author}(d)$ is unlikely to hold, since we know (almost surely) that d is a novel. Essentially, we lack the common-sense knowledge that

(i) anyone who has published a novel is an author, and

(ii) authors and novels are disjoint entities,

which helps us to distinguish such queries. Observe that (i) is a positive axiom and would lead to higher probabilities, whereas (ii) is a negative (constraining) axiom and would entail lower probabilities for some queries.

This problem has been widely studied in the context of classical databases under the name of *ontology-based data access* (OBDA) (Poggi et al. 2008), a popular paradigm that encodes the domain knowledge through an ontology, thus being able to deduce facts not explicitly specified in the database. Following this, we encode the domain knowledge using a Datalog$^\pm$ ontology (Calì, Gottlob, and Lukasiewicz 2012), which helps to break down the symmetries between open tuples, letting us distinguish more queries by comparing their upper and lower probability values.

We study the semantic and computational properties of OpenPDBs under Datalog$^\pm$ programs. The main distinction between a PDB and an OpenPDB is that the latter represents a set of probability distributions instead of a single one, and introduces the difficulty of choosing the distribution that will maximize (or minimize) the probability of a query. It is known that the data complexity of probabilistic UCQ evaluation in OpenPDBs exhibits the same dichotomy between P and PP as in PDBs for unions of conjunctive queries (Dalvi and Suciu 2012; Ceylan, Darwiche, and Van den Broeck 2016). We lift this dichotomy to first-order rewritable (positive) Datalog$^{\pm}$ programs using standard techniques. We then show that, once negative constraints are allowed, reasoning can become $\mathrm{NP}^{\mathrm{PP}}$-hard. This result demonstrates the difference between OpenPDBs and PDBs, as in the latter reasoning with ontologies remains in PP.

We also propose an approximating semantics that circumvents the increase in complexity caused by negative constraints, and lift the dichotomy to general first-order rewritable programs under this semantics. We conclude with complexity results beyond the data complexity for ontology-mediated query evaluation relative to (tuple-independent) PDBs and OpenPDBs.

2 Background and Motivation

We briefly recall the basics of tuple-independent PDBs and their open-world variant OpenPDBs. We then highlight the advantages of accessing probabilistic data through a logical theory and provide an overview of Datalog$^{\pm}$ programs.

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

Queries and Databases. A *conjunctive query (CQ)* over γ is an existentially quantified formula $\exists \mathbf{x}\, \phi$, where ϕ is a conjunction of γ-atoms, written as a comma-separated list. A *union of conjunctive queries (UCQ)* is a disjunction of CQs. A query is *Boolean* if it has no free variables. A database $\mathcal{D}$ over γ is a finite set of γ-tuples. The central problem studied for databases is *query evaluation*: Finding all *answers* to a query Q over a database $\mathcal{D}$, which are assignments of the free variables in Q to constants such that the resulting first-order formula is satisfied in $\mathcal{D}$ in the usual sense, i.e., there is a homomorphism from the atoms in Q to the tuples in $\mathcal{D}$. In the following, we consider only Boolean queries Q, and focus on the associated decision problem, i.e., deciding whether Q is satisfied in $\mathcal{D}$, denoted as usual by $\mathcal{D} \vDash \mathsf{Q}$.

Example 1. Consider the database $\mathcal{D}_{ex} := \{\mathtt{Author}(a),\ \mathtt{Pub}(a,b),\ \mathtt{Pub}(c,d),\ \mathtt{Novel}(d)\}$ and the Boolean query $\mathsf{Q}_1 := \exists x_1, x_2\, \mathtt{Author}(x_1),\ \mathtt{Pub}(x_1,x_2)$.[1] Then, $\mathcal{D}_{ex} \vDash \mathsf{Q}_1$, since $\{\mathtt{Author}(a),\ \mathtt{Pub}(a,b)\} \vDash \mathsf{Q}_1$.

Probabilistic Databases. The most elementary probabilistic database model is based on the tuple-independence assumption. We adopt this model and refer to (Suciu et al. 2011) for details on this model and alternatives. A probabilistic database induces a set of classical databases (called *worlds*), each of which is associated with a probability value.

Formally, a *probabilistic database (PDB)* $\mathcal{P}$ over γ is a finite set of *(probabilistic) tuples* of the form $\langle t : p \rangle$, where t is a γ-tuple and $p \in [0,1]$, and, whenever $\langle t : p \rangle, \langle t : q \rangle \in \mathcal{P}$, then $p = q$. A PDB $\mathcal{P}$ assigns, to every γ-tuple t, the probability p, if $\langle t : p \rangle \in \mathcal{P}$, and the probability 0, otherwise.

Under the *tuple-independence* assumption, any such probability assignment P induces the following *unique joint probability distribution* over classical databases $\mathcal{D}$:

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

Accordingly, query evaluation is enriched to also consider the probabilistic information. More formally, the *probability of a Boolean query* Q w.r.t. P is $\mathrm{P}(\mathsf{Q}) := \sum_{\mathcal{D} \vDash \mathsf{Q}} \mathrm{P}(\mathcal{D})$. Here, we do not need to consider worlds with probability 0; e.g., if $\mathrm{P}(t) = 0$, then the worlds containing t do not affect $\mathrm{P}(\mathsf{Q})$.

Example 2. Consider the PDB $\mathcal{P}_{ex}$ from the introduction and Q_1 from Example 1. The probability of Q_1 on $\mathcal{P}_{ex}$ is obtained by summing the probabilities of the worlds that satisfy Q_1, i.e., all worlds containing the first two tuples, resulting in the probability 0.48. In contrast, the natural query

$$\mathsf{Q}_2 := \exists x_1, x_2\, \mathtt{Author}(x_1),\ \mathtt{Pub}(x_1,x_2),\ \mathtt{Novel}(x_2)$$

evaluates to 0 on $\mathcal{P}_{ex}$, since all worlds that satisfy this query have probability 0.

Open-World Probabilistic Databases. An *open-world probabilistic database (OpenPDB)* over γ is a pair $\mathcal{G} = (\mathcal{P}, \lambda)$, where $\lambda \in [0,1]$ and $\mathcal{P}$ is a PDB. A λ-*completion* of $\mathcal{G}$ is a PDB that is obtained by introducing, for each γ-tuple t that does not occur in $\mathcal{P}$ (called an *open tuple*), a probabilistic tuple $\langle t : p \rangle$ with $p \in [0, \lambda]$. For a fixed value $\alpha \in [0, \lambda]$, we define a special λ-completion, denoted $\mathcal{P}_\alpha$, in which the probabilities of all open tuples are equal to α. Note that $\mathcal{P}_0$ is equivalent to $\mathcal{P}$.

Example 3. Consider the OpenPDB $\mathcal{G}_{ex} := (\mathcal{P}_{ex}, 0.5)$. The set $\mathcal{P}_{ex} \cup \{\langle \mathtt{Novel}(b) : 0.2 \rangle\}$ is a λ-completion of $\mathcal{G}_{ex}$ (tuples with probability 0 are omitted).

An OpenPDB $\mathcal{G} = (\mathcal{P}, \lambda)$ defines the set $\mathrm{K}_{\mathcal{G}}$ of all probability distributions P induced by the λ-completions of $\mathcal{G}$. $\mathrm{K}_{\mathcal{G}}$ constitutes a so-called *credal set*, which means that it is closed, convex, and has a finite number of extremal points (Cozman 2000). The range of probabilities of a query under such a set can be expressed as a probability interval. Formally, the *probability interval* of a Boolean query Q w.r.t. $\mathcal{G}$ is $\mathrm{K}_{\mathcal{G}}(\mathsf{Q}) := [\underline{\mathrm{P}}_{\mathcal{G}}(\mathsf{Q}), \overline{\mathrm{P}}_{\mathcal{G}}(\mathsf{Q})]$, where

$$\underline{\mathrm{P}}_{\mathcal{G}}(\mathsf{Q}) := \min_{\mathrm{P} \in \mathrm{K}_{\mathcal{G}}} \mathrm{P}(\mathsf{Q}) \quad \text{and} \quad \overline{\mathrm{P}}_{\mathcal{G}}(\mathsf{Q}) := \max_{\mathrm{P} \in \mathrm{K}_{\mathcal{G}}} \mathrm{P}(\mathsf{Q}).$$

Example 4. Consider again the OpenPDB $\mathcal{G}_{ex}$. While the lower probability $\underline{\mathrm{P}}_{\mathcal{G}}(\mathsf{Q}_2)$ remains 0, the upper probability evaluates to $\overline{\mathrm{P}}_{\mathcal{G}}(\mathsf{Q}_2) > 0$ due to the λ-completion

$$\mathcal{P}_{0.5} = \mathcal{P}_{ex} \cup \{\langle \mathtt{Author}(b) : 0.5 \rangle, \langle \mathtt{Author}(c) : 0.5 \rangle, \ldots\},$$

which contains all open tuples with probability $\lambda = 0.5$.

This example shows that OpenPDBs improve our view of the domain compared to PDBs. However, we have already illustrated in the introduction that OpenPDBs can further benefit from an axiomatic encoding of the domain knowledge, since many queries involving open tuples will yield the same lower and upper probabilities, although according to common-sense knowledge, they should differ. This motivates our introduction of a logical theory, in the form of Datalog$^{\pm}$ rules, to formalize such knowledge.

[1] For ease of presentation, we assume that γ consists of the symbols appearing in the database and query (and later in the program).

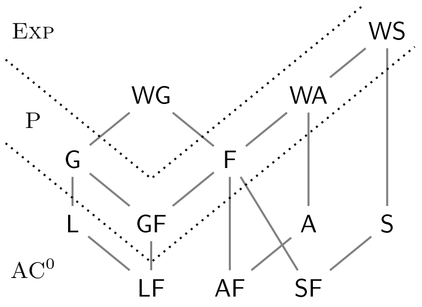

Figure 1: Inclusion relations and data complexity of UCQ entailment for Datalog$^\pm$ languages (Lukasiewicz et al. 2015)

Datalog$^\pm$ Programs. We now extend the vocabulary γ by a (potentially infinite) set $\mathbf{N}$ of *nulls*. An *instance* I over γ is a (possibly infinite) set of γ-tuples that may additionally contain nulls.

A *tuple-generating dependency (TGD)* σ is a first-order formula $\forall \mathbf{x}\, \varphi(\mathbf{x}) \rightarrow \exists \mathbf{y}\, \mathtt{P}(\mathbf{x}, \mathbf{y})$, where $\varphi(\mathbf{x})$ is a conjunction of γ-atoms, called the *body* of σ, and $\mathtt{P}(\mathbf{x}, \mathbf{y})$ is a γ-atom, called the *head* of σ. A *negative constraint (NC)* ν is a first-order formula $\forall \mathbf{x}\, \varphi(\mathbf{x}) \rightarrow \bot$, where $\varphi(\mathbf{x})$ is a conjunction of γ-atoms, called the *body* of ν, and $\bot$ is the truth constant *false*. A *(Datalog$^\pm$) program* Σ is a finite set of TGDs and NCs.[2] An *ontology-mediated query (OMQ)* is a pair (Q, Σ), where Σ is a program, and Q is a Boolean query.

An instance I satisfies a TGD or NC σ, if $I \models \sigma$, where $\models$ denotes the standard first-order entailment relation. I satisfies a program Σ, written $I \models \Sigma$, if I satisfies each formula in Σ. The set of *models* of a program Σ relative to a database $\mathcal{D}$, denoted $\mathit{mods}(\mathcal{D}, \Sigma)$, is $\{I \mid I \supseteq \mathcal{D} \text{ and } I \models \Sigma\}$. $\mathcal{D}$ is *consistent* w.r.t. Σ, if $\mathit{mods}(\mathcal{D}, \Sigma)$ is non-empty. The OMQ (Q, Σ) is *entailed* by $\mathcal{D}$, denoted $\mathcal{D} \models (\mathsf{Q}, \Sigma)$, if $I \models \mathsf{Q}$ holds for all $I \in \mathit{mods}(\mathcal{D}, \Sigma)$.

In general, the entailment problem is undecidable (Beeri and Vardi 1981). For this reason, many different restrictions on the TGDs have been proposed. We consider here *guarded* (G), *linear* (L), *sticky* (S), *acyclic* (A), *weakly guarded* (WG), *weakly sticky* (WS), and *weakly acyclic* (WA) sets of TGDs (Calì, Gottlob, and Kifer 2013; Calì, Gottlob, and Pieris 2012). Other important classes are given by *full* TGDs (F), *full and guarded* TGDs (GF), and similarly for LF, SF, and AF. Figure 1 illustrates the inclusion relations between these classes; for a more detailed description, see the extended version of this paper. We extend all these notions to programs Σ in the obvious way; for instance, Σ is guarded if all the TGDs in Σ are guarded. In the following, we use $\mathcal{L}$ to denote the set of Datalog$^\pm$ languages introduced above.

A key paradigm in OBDA is the *FO-rewritability* of queries; an OMQ (Q, Σ) is *FO-rewritable*, if there exists a Boolean UCQ Q_Σ such that, for all databases $\mathcal{D}$ that are consistent w.r.t. Σ, we have $\mathcal{D} \models (\mathsf{Q}, \Sigma)$ iff $\mathcal{D} \models \mathsf{Q}_\Sigma$. In this case, Q_Σ is called a *FO-rewriting* of (Q, Σ). A class of programs X is *FO-rewritable*, if it admits an FO-rewriting for any UCQ and program in X; these classes are characterized by a data complexity of AC^0 (see Figure 1).

3 Ontology-Mediated Queries for OpenPDBs

We now introduce the basics of OMQ evaluation relative to OpenPDBs. In the following, we assume that the input PDB $\mathcal{P}$ induces a consistent distribution w.r.t. the program. Formally, a probability distribution P is *consistent* w.r.t. Σ, if the database $\{t \mid \mathrm{P}(t) > 0\}$ is consistent w.r.t. Σ. Note that this assumption does not change the nature of the problem. The semantics of OMQs is again based on λ-completions. The difference appears in the deductive power provided by the Datalog$^\pm$ program, which is taken into consideration in the query semantics.

Definition 5 (Semantics)**.** The *probability of an OMQ* (Q, Σ) *relative to a probability distribution* P is

$$\mathrm{P}(\mathsf{Q}, \Sigma) = \sum_{\mathcal{D} \models (\mathsf{Q}, \Sigma)} \mathrm{P}(\mathcal{D}),$$

where $\mathcal{D}$ ranges over all databases over γ. The *probability interval of* (Q, Σ) *relative to an OpenPDB* $\mathcal{G}$ is then given by $\mathrm{K}_\mathcal{G}(\mathsf{Q}, \Sigma) := [\underline{\mathrm{P}}_\mathcal{G}(\mathsf{Q}, \Sigma), \overline{\mathrm{P}}_\mathcal{G}(\mathsf{Q}, \Sigma)]$, where

$$\underline{\mathrm{P}}_\mathcal{G}(\mathsf{Q}, \Sigma) := \min_{\mathrm{P} \in \mathrm{K}_\mathcal{G}} \{\mathrm{P}(\mathsf{Q}, \Sigma) \mid \mathrm{P} \text{ is consistent w.r.t. } \Sigma\},$$

$$\overline{\mathrm{P}}_\mathcal{G}(\mathsf{Q}, \Sigma) := \max_{\mathrm{P} \in \mathrm{K}_\mathcal{G}} \{\mathrm{P}(\mathsf{Q}, \Sigma) \mid \mathrm{P} \text{ is consistent w.r.t. } \Sigma\}.$$

The special case of $\lambda = 0$ corresponds to having a single (closed-world) PDB $\mathcal{P}$. In this case, we simply speak of the *probability of* (Q, Σ) *relative to a PDB* $\mathcal{P}$.

This semantics defers the decision of whether a world satisfies a query to an entailment test. However, we maximize only over consistent λ-completions, i.e., the ones that induce consistent distributions, which is the most important aspect of this semantics.

3.1 Semantic Considerations

In the following, we evaluate our semantics w.r.t. the goals identified in the motivation of this paper, and discuss our choice of restricting to the consistent λ-completions.

Distinguishing Queries. We argued that OpenPDBs can benefit from an axiomatic encoding of the knowledge of the domain. Consider again our running example, which is now enriched with a program.

Example 6. Consider the OpenPDB $\mathcal{G}_{ex}$ given before and the program $\Sigma_{ex} := \{\mathtt{Author}(x), \mathtt{Novel}(x) \rightarrow \bot, \mathtt{Pub}(x, y), \mathtt{Novel}(y) \rightarrow \mathtt{Author}(x)\}$ which states that authors and novels are disjoint entities, and that anyone who has published a novel is an author. The lower probability of $\mathtt{Author}(d)$ remains 0, while the upper probability is now reduced to 0 with the help of the program Σ_{ex}. In contrast, the lower probability of $\mathtt{Author}(c)$ increases to 0.9, while the upper probability increases to 0.95. These intervals are much more informative than the default interval $[0, 0.5]$.

[2] For brevity, we omit the universal quantifiers in TGDs and NCs, and use commas (instead of $\wedge$) for conjoining atoms. For clarity, we consider single-atom-head TGDs; however, our results can be easily extended to TGDs with conjunctions of atoms in the head.

Restricting to Consistent Distributions. The most subtle aspect of choosing the *best* distribution is the question of how to deal with inconsistent worlds. Ignoring inconsistencies (and optimizing over *all* completions) leads to a drowning effect: since inconsistent worlds entail everything, this semantics would be biased towards choosing inconsistent λ-completions. This does not satisfy our goals, as even an unsatisfiable query could evaluate to a positive probability.

An alternative approach, which is standard for (closed-world) PDBs, and is quite intuitive at first glance, would be to choose the distribution which maximizes the conditional probability $\mathrm{P}((\mathsf{Q},\Sigma) \mid (\mathcal{D},\Sigma) \not\models \bot)$, i.e., the probability of the query on the set of all consistent worlds. A careful inspection, however, shows that this semantics also favors inconsistent distributions over consistent ones. To illustrate this, consider our running example, and suppose that we want to compute the upper probability of Q_2 (mediated by Σ_{ex}). The semantics based on the conditional probability would favor the λ-completion $\mathcal{P}_{0.5}$, even though this PDB is highly inconsistent. This is mainly due to the normalization process internal to the computation. As part of this normalization, the probability mass of inconsistent worlds is distributed to consistent worlds. As a consequence, it is often possible to increase the query probability by simply increasing the probability of inconsistent worlds. This is not a desired effect, since we are interested in finding the most suitable λ-completion from the open world, and not the one that increases the query probability by increasing the probability mass of inconsistent worlds.

To avoid such drowning effects, our proposal considers only consistent distributions. That is, we do not want to introduce inconsistencies when completing our knowledge over the domain by choosing a λ-completion. One drawback of our approach is the fact that inconsistencies are not tolerated even if the inconsistency degree is very small. However, it would be easy to introduce a threshold value, say 0.1, to tolerate the inconsistent completions where the probability of the inconsistent worlds does not exceed this threshold.

4 Data Complexity Results

We now formulate the task of probabilistic query evaluation as a decision problem.

Definition 7 (Decision Problems)**.** Let (Q,Σ) be an OMQ, $\mathcal{G}$ an OpenPDB and $p \in [0,1]$. The problem of *upper (resp., lower) probabilistic query entailment* is to decide whether $\overline{\mathrm{P}}_{\mathcal{G}}(\mathsf{Q},\Sigma) > p$ (resp., $\underline{\mathrm{P}}_{\mathcal{G}}(\mathsf{Q},\Sigma) < p$) holds. *Probabilistic query entailment relative to PDBs* is a special case, where $\lambda = 0$.

Note that this definition is rather general, but in the scope of this paper, we are concerned with UCQs, and thus we use the term *probabilistic UCQ entailment* instead. Moreover, we are mainly concerned with the *data complexity*, which is calculated based on the size of the OpenPDB; i.e., the schema $\mathbf{R}$, the query Q, and the program Σ are assumed to be fixed (Vardi 1982). The relevant data complexity results for UCQ entailment in Datalog$^{\pm}$ are summarized in Figure 1.

Most of our complexity results are related to the complexity class PP (Gill 1977), which comprises the languages recognized by a polynomial-time non-deterministic Turing machine that accepts an input if and only if *more than half* of the computation paths are accepting (Torán 1991). Intuitively, PP is the decision counterpart of #P (Valiant 1979). For details on the complexity classes used in our results, and the types of reductions, we refer to the extended version of this paper. It has been shown in (Dalvi and Suciu 2012) that probabilistic UCQ entailment for PDBs exhibits a dichotomy between P and PP. Queries that admit a P algorithm are called *safe* and the remaining ones *unsafe*. This result has been lifted to OpenPDBs in (Ceylan, Darwiche, and Van den Broeck 2016). For detailed insights on the class of safe queries, we refer to the original papers. The CQ $\exists x,y\, \mathsf{C}(x) \wedge \mathsf{L}(x,y) \wedge \mathsf{S}(y)$ is the prototypical example of an unsafe query; it is connected and can not be decomposed into independent queries in an efficient manner (applying certain rules from (Dalvi and Suciu 2012)). However, removing any of the atoms from this query makes it safe.

We borrow this notion, and say that an OMQ (Q,Σ) is *safe*, if there exist polynomial-time algorithms for lower and upper probabilistic entailment of (Q,Σ) relative to any OpenPDB (resp., PDB).

4.1 Positive Programs

We first consider *positive* Datalog$^{\pm}$ programs, which do not contain NCs. Under this restriction, there are no inconsistent distributions, and Definition 5 simplifies. We later show that this distinction is important, since the complexity increases in the presence of NCs. This is surprising, as in the classical case NCs are usually not problematic.

Recall that OpenPDBs induce an infinite set of probability distributions that form a credal set, which has the following useful property (Cozman 2000): To determine the upper or lower probability of an event, it suffices to consider the *extremal* probability distributions, which are obtained by setting the probability values of all elementary events to one of the extreme points. In the context of OpenPDBs, this means that each of the open tuples may have probability λ or 0, but no intermediate choices need to be examined. For UCQs, this implies an even stronger result.

Lemma 8. *Let (Q,Σ) be an OMQ, where Q is a UCQ and Σ is a positive Datalog$^{\pm}$ program. Then, it holds that* $\mathrm{K}_{\mathcal{G}}(\mathsf{Q},\Sigma) = [\mathrm{P}_{\mathcal{P}_0}(\mathsf{Q},\Sigma), \mathrm{P}_{\mathcal{P}_\lambda}(\mathsf{Q},\Sigma)]$.

Thus, it suffices to consider a single λ-completion (either $\mathcal{P}_0$ or $\mathcal{P}_\lambda$) and the particular distribution it induces. As a result, probabilistic UCQ entailment can be solved by standard methods; i.e., summing up the probabilities of all worlds that pass the entailment test. This naïve approach yields tight complexity bounds for the considered problems.

Theorem 9. *Probabilistic UCQ entailment is* PP*-complete for the languages in $\mathcal{L} \setminus \{\mathsf{WG}\}$; it is* Exp*-complete in* WG.

This result is of no surprise given the PP-hardness of inference in OpenPDBs. However, all our PP-hardness results are based on the result of (Dalvi and Suciu 2012), and hence are valid only with respect to Turing reductions. All other complexity results in this paper also hold under standard many-one reductions. It is an open problem to find a UCQ for which probabilistic entailment is PP-hard w.r.t. many-one reductions. The striving question is now whether it is possible to lift the dichotomy result from OpenPDBs. For this purpose, we elaborate on query rewritability.

Lemma 10. *Let* $(\mathtt{Q},\Sigma)$ *be an OMQ,* P *be a tuple-independent probability distribution over worlds such that* $\mathrm{P}(\mathcal{D})=0$ *whenever* $\mathcal{D}$ *is inconsistent w.r.t.* Σ*, and* $\mathtt{Q}_\Sigma$ *be an FO-rewriting of* $(\mathtt{Q},\Sigma)$*. Then, we have* $\mathrm{P}(\mathtt{Q},\Sigma)=\mathrm{P}(\mathtt{Q}_\Sigma)$.

Since all worlds are consistent under positive programs, Lemmas 8 and 10 imply that we can reduce probabilistic UCQ entailment under positive programs to the case of OpenPDBs via query rewriting.

Corollary 11. *Let* $(\mathtt{Q},\Sigma)$ *be an OMQ, where* $\mathtt{Q}$ *is a UCQ, and* Σ *is a positive program, and* $\mathtt{Q}_\Sigma$ *be an FO-rewriting of* $(\mathtt{Q},\Sigma)$*. Then, for any OpenPDB* $\mathcal{G}$*, it holds that* $\overline{\mathrm{P}}_{\mathcal{G}}(\mathtt{Q},\Sigma)=\overline{\mathrm{P}}_{\mathcal{G}}(\mathtt{Q}_\Sigma)$ *and* $\underline{\mathrm{P}}_{\mathcal{G}}(\mathtt{Q},\Sigma)=\underline{\mathrm{P}}_{\mathcal{G}}(\mathtt{Q}_\Sigma)$.

We now obtain a dichotomy from the results in (Dalvi and Suciu 2012; Ceylan, Darwiche, and Van den Broeck 2016).

Theorem 12. *Let* $(\mathtt{Q},\Sigma)$ *be an OMQ , where* $\mathtt{Q}$ *is a UCQ, and* Σ *is a positive program, and* $\mathtt{Q}_\Sigma$ *be a rewriting of* $(\mathtt{Q},\Sigma)$*. Then,* $(\mathtt{Q},\Sigma)$ *is safe iff* $\mathtt{Q}_\Sigma$ *is safe (over OpenPDBs). If* $(\mathtt{Q},\Sigma)$ *is not safe, then it is* PP*-hard.*

In particular, either all rewritings of $(\mathtt{Q},\Sigma)$ are safe, or none of them are. Hence, in FO-rewritable languages, we can take an *arbitrary* rewriting and check safety using the characterization of (Dalvi and Suciu 2012). Such a rewriting can be obtained by well-known algorithms, e.g., using backward chaining of TGDs (Gottlob, Orsi, and Pieris 2011).

To conclude this section, we illustrate some effects that simple positive programs can have on the complexity of probabilistic query entailment.

Example 13. The query $\exists x,y\,\mathtt{C}(x)\wedge\mathtt{M}(x,y)$ is safe for OpenPDBs. It becomes unsafe under the TGD $\mathtt{R}(x,y),\mathtt{T}(y)\rightarrow\mathtt{M}(x,y)$, since then it rewrites to the query $(\exists x,y\,\mathtt{C}(x),\mathtt{M}(x,y))\vee(\exists x,y\,\mathtt{C}(x),\mathtt{R}(x,y),\mathtt{T}(y))$. Conversely, the CQ $\exists x,y\,\mathtt{C}(x)\wedge\mathtt{L}(x,y)\wedge\mathtt{S}(y)$ is not safe for OpenPDBs, but becomes safe under $\mathtt{L}(x,y)\rightarrow\mathtt{S}(y)$, as it rewrites to $\exists x,y\,\mathtt{C}(x)\wedge\mathtt{L}(x,y)$. Note that these are very simple TGDs, which are full, acyclic, guarded, and sticky.

4.2 Programs with Negative Constraints

In the presence of NCs, it still suffices to consider the extremal λ-completions. In fact, once the correct completion is known, the probabilistic UCQ entailment problem can still be reduced to probabilistic inference (in FO-rewritable languages). The key difference in the presence of NCs is that we have to make sure that this completion is consistent. That is, choosing the completion $\mathcal{P}_\lambda$ that sets all open tuples to λ (as in Lemma 8) is not feasible, as this will very likely lead to inconsistencies. However, observe that the *lower* probability can still be obtained from the completion $\mathcal{P}_0$ (which we assumed to be consistent), and hence the previous results still hold for lower probabilistic UCQ entailment with NCs.

A naïve way of solving the upper probabilistic UCQ entailment problem is to *guess* a λ-completion and then check whether it is consistent and compare the resulting probability to the threshold. This yields an $\mathrm{NP}^{\mathrm{PP}}$ upper bound for our decision problem. Our next result shows a matching lower bound for the class GF, and so for all considered Datalog$^\pm$ languages with data complexity above AC^0 (see Figure 1).

Theorem 14. *Upper probabilistic UCQ entailment is* $\mathrm{NP}^{\mathrm{PP}}$*-complete in full, guarded programs. It is* PP*-complete for all languages with polynomial data complexity once restricted to PDBs.*

This result is by reduction from the $\mathrm{NP}^{\mathrm{PP}}$-complete problem of finding a partial assignment for designated variables of a propositional formula in CNF, for which the number of satisfying assignments extending this partial assignment is above some threshold (Wagner 1986). On the one hand, this result is surprising, as NCs are not problematic for PDBs, even with normalization semantics; on the other hand, this is not so surprising, as non-monotonicity is also a source of additional hardness in OpenPDBs: query evaluation becomes $\mathrm{NP}^{\mathrm{PP}}$-complete in OpenPDBs if negated atoms are allowed in UCQs (Ceylan, Darwiche, and Van den Broeck 2016). In contrast, our result applies to UCQs without negated atoms, and thus it is much more involved. The proof encodes the non-determinism into the NCs, which are not as powerful as non-monotone queries, and uses TGDs to check the satisfaction condition of the clauses in the CNF.

Before concluding this section, we illustrate the effects of NCs on some examples, which also show the difficulties in lifting the dichotomy of Theorem 12 to NCs.

Example 15. Consider the query $(\exists x,y\,\mathtt{C}(x)\wedge\mathtt{S}(y))\vee(\exists x,y\,\mathtt{C}(x)\wedge\mathtt{L}(x,y))$, which is not safe for OpenPDBs, but becomes safe relative to the NC $\mathtt{S}(y),\mathtt{L}(x,y)\rightarrow\bot$. The reason is that the algorithm of (Dalvi and Suciu 2012) that decides safety will produce the unsafe query $\exists x,y\,\mathtt{C}(x)\wedge\mathtt{S}(y)\wedge\mathtt{L}(x,y)$ through a sequence of reduction rules; however, this query automatically has probability 0 under the given NC, and hence becomes trivially safe.

Approximations for Programs with NCs. Motivated by the high complexity of reasoning in programs with NCs, we propose an alternative semantics, which approximates the semantics of Definition 5. Observe that the upper probability $\overline{\mathrm{P}}_{\mathcal{G}}(\mathtt{Q},\Sigma)$ will always be obtained at a λ-completion that adds as many open tuples as possible to the original $\mathcal{P}$ without causing an inconsistency. This is related to the notion of a database *repair*, which is a maximal consistent subset of an inconsistent database (Arenas, Bertossi, and Chomicki 1999). Instead of considering all possible repairs, an easier alternative is to compute the intersection of all repairs and use this for query answering (Lembo et al. 2010). In our setting, however, we are not actually repairing an inconsistent initial database $\mathcal{P}$, but rather assume that all tuples in $\mathcal{P}$ are correct and consistent, and hence need to take care that no such tuples are removed in this intersection. Formally, given an OMQ $(\mathtt{Q},\Sigma)$ and an OpenPDB $\mathcal{G}=(\mathcal{P},\lambda)$, we consider the special λ-completion $\mathcal{P}_\cap$ that is constructed as the intersection of all $\subseteq$-maximal consistent subsets of $\mathcal{P}_\lambda$ that contain $\mathcal{P}$ (all tuples not in this intersection have probability 0).

Definition 16 (Intersection Semantics)**.** The *probability interval* of $(\mathtt{Q},\Sigma)$ relative to an OpenPDB

Datalog$^\pm$ Languages	**PDBs**		**OpenPDBs**	
	fs-**c.**	*fp*-**c.**	*fs*-**c.**	*fp*-**c.**
L, LF, AF	$\mathrm{PP}^{\mathrm{NP}}$	$\mathrm{PP}^{\mathrm{NP}}$	$\mathrm{NP}^{\mathrm{PP}}$	$\mathrm{NP}^{\mathrm{PP}}$
G	Exp	$\mathrm{PP}^{\mathrm{NP}}$	Exp	$\mathrm{NP}^{\mathrm{PP}}$
WG	Exp	Exp	Exp	Exp
S, F, SF, GF	$\mathrm{PP}^{\mathrm{NP}}$	$\mathrm{PP}^{\mathrm{NP}}$	$\mathrm{NP}^{\mathrm{PP}}$	$\mathrm{NP}^{\mathrm{PP}}$
A	NExp	$\mathrm{PP}^{\mathrm{NP}}$	in P^{NE}	$\mathrm{NP}^{\mathrm{PP}}$
WS, WA	2exp	$\mathrm{PP}^{\mathrm{NP}}$	2exp	$\mathrm{NP}^{\mathrm{PP}}$

Table 1: (fs/fp)-combined complexity of probabilistic UCQ entailment relative to OpenPDBs and PDBs.

$\mathcal{G} = (\mathcal{P}, \lambda)$ under the *intersection semantics* is defined as $\mathrm{K}^{\cap}_{\mathcal{P}}(\mathsf{Q}, \Sigma) := [\mathrm{P}_{\mathcal{P}_0}(\mathsf{Q}, \Sigma), \mathrm{P}_{\mathcal{P}_\cap}(\mathsf{Q}, \Sigma)]$.

As with positive programs (cf. Lemma 8), probabilistic UCQ entailment under this semantics is PP-complete in all Datalog$^\pm$ languages where classical UCQ entailment is in P. More interestingly, we can also show a dichotomy for FO-rewritable queries with the help of Lemma 10.

Theorem 17. *Let* (Q, Σ) *be an OMQ , where* Q *is a UCQ, and* Σ *is a program, and* Q_Σ *be a rewriting of* Q *relative to* Σ*. Then,* (Q, Σ) *is safe under intersection semantics iff* Q_Σ *is safe (over OpenPDBs). If* (Q, Σ) *is not safe under intersection semantics, then it is* PP*-hard.*

5 Beyond Data Complexity

For the sake of completeness, we also provide results beyond the data complexity. We consider *fixed-program combined (fp-combined) complexity*, which is calculated in the size of the database and the query, while the program and schema remain fixed. Additionally, we remove the assumption that the program is fixed, and study *fixed-schema combined (fs-combined) complexity*. Our results are summarized in Table 1; all results except one are completeness results. The results are given relative to both PDBs and OpenPDBs to emphasize the computational differences.

Theorem 18. *Let* X *be a class of programs, and UCQ entailment in* X *be* $\mathbf{C}$*-complete in (fs/fp)-combined complexity. Then, probabilistic UCQ entailment in* X *is* $\mathbf{C}$*-hard and in* $\mathrm{PSPACE}^{\mathbf{C}}$ *in (fs/fp)-combined complexity. If* $\mathbf{C} = \mathrm{NExp}$*, it is in* P^{NE}*, and* NExp*-complete when restricted to PDBs.*

Hence, if $\mathbf{C} = \mathrm{Exp}$ or $\mathbf{C} = 2\mathrm{EXP}$, the complexity is not affected by adding OpenPDBs, since the complexity of UCQ entailment dominates the problem. We now consider the special case of NP-complete classes.

Theorem 19. *Let* X *be a class of programs. If UCQ entailment in* X *is* NP*-complete in (fs/fp)-combined complexity, then probabilistic UCQ entailment in* X *is complete for* $\mathrm{NP}^{\mathrm{PP}}$ *in (fs/fp)-combined complexity; it is complete for* $\mathrm{PP}^{\mathrm{NP}}$ *when restricted to a PDB.*

The hardness proof uses no TGDs and only one NC. This implies that the additional hardness in probabilistic UCQ entailment relative to OpenPDBs is caused solely by the interaction between NCs and the open-world semantics. This provides more evidence that OpenPDBs with NCs are more powerful than PDBs with NCs.

6 Related Work

Our work builds on the research on probabilistic databases, which has a long tradition (Imieliski and Lipski 1984; Fuhr and Rölleke 1997; Suciu et al. 2011). We focus on tuple-independent probabilistic databases, with an emphasis on the dichotomy result of Dalvi and Suciu (2012). The most closely related work is by Jung and Lutz (2012), where the authors lift the dichotomy result of PDBs to the lightweight description logics $\mathcal{EL}$ and *DL-Lite* over PDBs; they even describe the case of an ontology language that is not FO-rewritable and causes all CQs of a certain form to become #P-hard. In contrast, we consider the more expressive languages of the Datalog$^\pm$ family and provide results both relative to PDBs and OpenPDBs. We show that the semantic differences between these formalisms lead to different results (even in the data complexity).

Most of the recent work on probabilistic query answering using ontologies is based on lightweight ontology languages. Some (D'Amato, Fanizzi, and Lukasiewicz 2008; Ceylan and Peñaloza 2015; Gottlob et al. 2013) result from a combination of ontologies with probabilistic graphical models such as Bayesian networks (Pearl 1988) or Markov logic networks (Richardson and Domingos 2006). Both the semantics and the assumptions used in these works are very different than ours. More closely related is the work by Ceylan, Peñaloza, and Lukasiewicz (2016), where the computational complexity of query answering in probabilistic Datalog$^\pm$ under the possible world semantics is investigated. Note, however, that the authors consider PDBs, and thus a unique probability distribution. Moreover, even for PDBs, the results are not comparable as they allow conditional dependencies and hence the hardness results do not apply to the special case of tuple-independent PDBs.

Possible world semantics is common in probabilistic logic programming and relational probabilistic models (Renkens et al. 2012; Kwiatkowska, Norman, and Parker 2002; Poole 1997). OpenPDBs extend this semantics to a (finite) open universe, and allow imprecise probabilities (Levi 1980) for tuples in this universe. The latter can be seen as analogous to extending Bayesian networks (Pearl 1988) to credal networks (Cozman 2000; De Campos and Cozman 2005). Our framework enriches OpenPDBs further by mediating the query with an ontology, where the query evaluation problem over a database is replaced with a logical entailment problem, allowing us to deduce implicitly encoded facts.

7 Summary and Outlook

We introduced a refinement of the recently proposed OpenPDBs, using Datalog$^\pm$ ontologies to express additional background knowledge, and lifted the dichotomy from (Dalvi and Suciu 2012; Ceylan, Darwiche, and Van den Broeck 2016) to all FO-rewritable languages for positive programs. We showed that NCs can increase the worst-case complexity, and proposed an approximating semantics circumventing the increase in the complexity. Additionally, we provided complexity results beyond the data complexity.

In future work, we want to determine whether it is possible to obtain a dichotomy result for programs with NCs for FO-rewritable Datalog$^\pm$ languages. Similarly, the question whether the P-complete languages admit a dichotomy when restricting to positive programs is left as future work. Note also that we assume a finite set of constants (as in OpenPDBs), but allow infinitely many unknown individuals (nulls). Dealing with distributions over infinitely many objects as in BLOG (Milch et al. 2005) is an important task, and a crucial part of future work.

Acknowledgments

This work is supported by the German Research Foundation (DFG) within the Collaborative Research Center SFB 912 HAEC and the Graduiertenkolleg RoSI (GRK 1907), and by the UK EPSRC grants EP/J008346/1, EP/L012138/1, EP/M025268/1, and EP/N510129/1.

References

Arenas, M.; Bertossi, L.; and Chomicki, J. 1999. Consistent query answers in inconsistent databases. In *Proc. of PODS*, 68–79. ACM.

Beeri, C., and Vardi, M. Y. 1981. The implication problem for data dependencies. In *Proc. of ICALP*, 73–85. Springer.

Calì, A.; Gottlob, G.; and Kifer, M. 2013. Taming the infinite chase: Query answering under expressive relational constraints. *JAIR* 48:115–174.

Calì, A.; Gottlob, G.; and Lukasiewicz, T. 2012. A general Datalog-based framework for tractable query answering over ontologies. *J. Web Sem.* 14:57–83.

Calì, A.; Gottlob, G.; and Pieris, A. 2012. Towards more expressive ontology languages: The query answering problem. *AIJ* 193:87–128.

Ceylan, İ. İ., and Peñaloza, R. 2015. Probabilistic query answering in the Bayesian description logic BEL. In *Proc. of SUM*, 21–35.

Ceylan, İ. İ.; Darwiche, A.; and Van den Broeck, G. 2016. Open-world probabilistic databases. In *Proc. of KR*. AAAI Press.

Ceylan, İ. İ.; Peñaloza, R.; and Lukasiewicz, T. 2016. Complexity results for probabilistic Datalog+/–. In *Proc. of ECAI*. IOS Press.

Cozman, F. G. 2000. Credal networks. *AIJ* 120(2):199–233.

Dalvi, N., and Suciu, D. 2012. The dichotomy of probabilistic inference for unions of conjunctive queries. *J. ACM* 59(6):1–87.

D'Amato, C.; Fanizzi, N.; and Lukasiewicz, T. 2008. Tractable reasoning with Bayesian description logics. In *Proc. of SUM*, 146–159. Springer.

De Campos, C. P., and Cozman, F. G. 2005. The inferential complexity of Bayesian and credal networks. In *Proc. of IJCAI*, 1313–1318. AAAI Press.

Dong, X. L.; Gabrilovich, E.; Heitz, G.; Horn, W.; Lao, N.; Murphy, K. P.; Strohmann, T.; Sun, S.; and Zhang, W. 2014. Knowledge Vault: A web-scale approach to probabilistic knowledge fusion. In *Proc. of SIGKDD*, 601–610. ACM.

Fagin, R.; Kolaitis, P. G.; Miller, R. J.; and Popa, L. 2005. Data exchange: Semantics and query answering. *TCS* 336(1):89–124.

Fuhr, N., and Rölleke, T. 1997. A probabilistic relational algebra for the integration of information retrieval and database systems. *ACM Trans. Inf. Systems* 15(1):32–66.

Gill, J. T. 1977. Computatonal complexity of probabilistic Turing machines. *SIAM J. on Computing* 6(4):675–695.

Gottlob, G.; Lukasiewicz, T.; Martinez, M. V.; and Simari, G. I. 2013. Query answering under probabilistic uncertainty in Datalog+/– ontologies. *Ann. Math. Artif. Intell.* 69(1):37–72.

Gottlob, G.; Orsi, G.; and Pieris, A. 2011. Ontological queries: Rewriting and optimization. In *Proc. of ICDE*, 2–13. IEEE Press.

Hemachandra, L. A. 1989. The strong exponential hierarchy collapses. *J. Comput. Syst. Sci.* 39(3):299–322.

Hoffart, J.; Suchanek, F. M.; Berberich, K.; and Weikum, G. 2013. YAGO2: A spatially and temporally enhanced knowledge base from Wikipedia. In *Proc. of IJCAI*, 3161–3165.

Imieliski, T., and Lipski, W. 1984. Incomplete information in relational databases. *J. ACM* 31(4):761–791.

Jung, J. C., and Lutz, C. 2012. Ontology-based access to probabilistic data with OWL QL. In *Proc. of ISWC*, 182–197. Springer.

Kwiatkowska, M.; Norman, G.; and Parker, D. 2002. PRISM: Probabilistic symbolic model checker. In *Proc. TOOLS*, 200–204.

Lembo, D.; Lenzerini, M.; Rosati, R.; Ruzzi, M.; and Savo, D. F. 2010. Inconsistency-tolerant semantics for description logics. In *Proc. of RR*, 103–117. Springer.

Levi, I. 1980. *The Enterprise of Knowledge*. MIT Press.

Lukasiewicz, T.; Martinez, M. V.; Pieris, A.; and Simari, G. I. 2015. From classical to consistent query answering under existential rules. In *Proc. of AAAI*, 40–45. AAAI Press.

Milch, B.; Marthi, B.; Russell, S.; Sontag, D.; Ong, D. L.; and Kolobov, A. 2005. BLOG: Probabilistic models with unknown objects. In *Proc. of IJCAI*, 1352–1359. Morgan Kaufmann.

Mitchell, T.; Cohen, W.; Hruschka, E.; Talukdar, P.; Betteridge, J.; Carlson, A.; Dalvi, B.; and Gardner, M. 2015. Never-ending learning. In *Proc. of AAAI*, 2302–2310. AAAI Press.

Pearl, J. 1988. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann.

Poggi, A.; Lembo, D.; Calvanese, D.; Giacomo, G. D.; Lenzerini, M.; and Rosati, R. 2008. Linking data to ontologies. *J. Data Sem.* 10:133–173.

Poole, D. 1997. The independent choice logic for modelling multiple agents under uncertainty. *AIJ* 94(1-2):7–56.

Renkens, J.; Shterionov, D.; Van den Broeck, G.; Vlasselaer, J.; Fierens, D.; Meert, W.; Janssens, G.; and De Raedt, L. 2012. ProbLog2: From probabilistic programming to statistical relational learning. In *Proc. of NIPS*, 1–5.

Richardson, M., and Domingos, P. 2006. Markov logic networks. *Mach. Learn.* 62(1-2):107–136.

Shin, J.; Wang, F.; Sa, C. D.; Zhang, C.; and Wu, S. 2015. Incremental knowledge base construction using DeepDive. In *Proc. of VLDB*.

Suciu, D.; Olteanu, D.; Ré, C.; and Koch, C. 2011. *Probabilistic Databases*. Morgan & Claypool.

Toda, S. 1989. On the Computational Power of PP and +P. In *Proc. of SFCS*, 514–519. IEEE.

Torán, J. 1991. Complexity classes defined by counting quantifiers. *J. ACM* 38(3):753–774.

Valiant, L. G. 1979. The complexity of computing the permanent. *TCS* 8(2):189–201.

Vardi, M. Y. 1982. The complexity of relational query languages. In *Proc. of STOC*, 137–146.

Wagner, K. W. 1986. The complexity of combinatorial problems with succinct input representation. *Acta Inf.* 23(3):325–356.

Wu, W.; Li, H.; Wang, H.; and Zhu, K. Q. 2012. Probase: A probabilistic taxonomy for text understanding. In *Proc. of SIGMOD*, 481–492. ACM.

A Decidable Classes of TGDs

The most important (syntactic) restrictions on TGDs studied in the literature are guardedness (Calì, Gottlob, and Kifer 2013), stickiness (Calì, Gottlob, and Pieris 2012) and acyclicity, along with their "weak" counterparts, weak guardedness (Calì, Gottlob, and Kifer 2013), weak stickiness (Calì, Gottlob, and Pieris 2012), and weak acyclicity (Fagin et al. 2005), respectively.

A TGD is *guarded*, if there exists a body atom that contains (or "guards") all body variables. The class of guarded TGDs, denoted G, is defined as the family of all possible sets of guarded TGDs. A key subclass of guarded TGDs are the linear TGDs with just one body atom, which is automatically the guard. The class of linear TGDs is denoted by L. *Weakly guarded* TGDs extend guarded TGDs by requiring only the body variables that are considered "harmful" to appear in the guard (see (Calì, Gottlob, and Kifer 2013) for full details). The associated class of TGDs is denoted WG. It is easy to verify that $\mathsf{L} \subset \mathsf{G} \subset \mathsf{WG}$.

Stickiness is inherently different from guardedness, and its central property can be described as follows: variables that appear more than once in a body (i.e., join variables) must always be propagated (or "stuck") to the inferred atoms. A TGD that enjoys this property is called *sticky*, and the class of sticky TGDs is denoted by S. Weak stickiness generalizes stickiness by considering only "harmful" variables, and defines the class WS of *weakly sticky* TGDs. Observe that $\mathsf{S} \subset \mathsf{WS}$.

A set Σ of TGDs is *acyclic* (and belongs to the class A), if its predicate graph is acyclic. Equivalently, an acyclic set of TGDs can be seen as a non-recursive set of TGDs. Σ is *weakly acyclic*, if its dependency graph enjoys a certain acyclicity condition, which guarantees the existence of a finite canonical model; the associated class is denoted WA. Clearly, $\mathsf{A} \subset \mathsf{WA}$. Interestingly, it also holds that $\mathsf{WA} \subset \mathsf{WS}$ (Calì, Gottlob, and Pieris 2012).

Another key fragment of TGDs which deserves our attention are *full* TGDs, i.e., TGDs without existentially quantified variables. The corresponding class is denoted by F. Restricting full TGDs to satisfy linearity, guardedness, stickiness, or acyclicity yields the classes LF, GF, SF, and AF, respectively. It is known that $\mathsf{F} \subset \mathsf{WA}$ (Fagin et al. 2005) and $\mathsf{F} \subset \mathsf{WG}$ (Calì, Gottlob, and Kifer 2013).

B Complexity Classes

Throughout the paper, we use the standard assumption that the probability values are rational. The central complexity class for our analysis is the complexity class PP (Gill 1977), which defines the set of languages recognized by a polynomially time-bounded non-deterministic Turing machine that accepts an input if and only if *more than half* of the computation paths are accepting (Torán 1991). Intuitively, the class PP can be seen as the decision counterpart of #P (Valiant 1979). In fact, it is known that $\mathrm{P}^{\mathrm{PP}} = \mathrm{P}^{\#\mathrm{P}}$ (Toda 1989). Moreover, in (Toda 1989) it is also shown that $\mathrm{PP}^{\mathrm{PH}} \subseteq \mathrm{P}^{\mathrm{PP}}$, and hence we have $\mathrm{NP}^{\mathrm{PP}^{\mathrm{PH}}} = \mathrm{NP}^{\mathrm{P}^{\mathrm{PP}}} = \mathrm{NP}^{\mathrm{PP}}$.

Observe that the original dichotomy by Dalvi and Suciu (2012) is formulated using the class #P. We adopt the view of Ceylan, Darwiche, and Van den Broeck (2016), and consider the associated decision complexity class PP. Note however, that #P-hardness is shown as usual using FP-Turing reductions, which translates to a P versus PP dichotomy *under polynomial-time Turing reductions*, for the associated decision problem. All our results except the PP-hardness results hold even under standard many-one reductions. It is an open problem to find a UCQ for which probabilistic query entailment is PP-hard w.r.t. many-one reductions.

Some of the complexity classes relevant to our results relate to standard classes as follows:

$$\mathrm{NP} \quad \subseteq \quad \mathrm{PP} \quad \subseteq \quad \mathrm{PP}^{\mathrm{NP}},\ \mathrm{NP}^{\mathrm{PP}} \quad \subseteq \quad \textsc{Pspace}$$

C Proof Sketch of Lemma 8

Since Q contains no negations, switching the probability of an open tuple from 0 to λ cannot decrease the query probability $\mathrm{P}(\mathsf{Q}, \Sigma)$: even if this decreases the probability of the previous worlds that entail Q and have non-zero probability by a factor of $(1 - \lambda)$, for each of these worlds, there is now an additional world that entails Q and has non-zero probability with a corresponding factor of λ, which makes up for the loss. Additionally, adding a new tuple may add new worlds with non-zero probability that entail Q w.r.t. Σ. Hence, the minimal (maximal) query probability is obtained in the completion that contains the minimal (maximal) number of open tuples with probability λ. □

D Proof Sketch of Theorem 9

WG: By Lemma 8, it suffices to consider the probability distribution P induced by a special λ-completion (which can be constructed efficiently). Consider an algorithm that enumerates all possible worlds $\mathcal{D}$ that succeed on the entailment test and sums up their probabilities $\mathrm{P}(\mathcal{D})$. This algorithm runs in exponential time. Hardness follows from Exp-hardness of classical UCQ entailment in WG, which corresponds to probabilistic UCQ entailment where $\lambda = 0$ and all probabilities are 1.

$\mathcal{L} \setminus \{\mathsf{WG}\}$: We need to consider only a single probability distribution P. We now create multiples of each world (which then correspond to the nondeterministic branches of a Turing machine M), in such a way that the uniform distribution over all thus generated worlds is equivalent to P when each copy is taken to represent its original world. Then, for thresholds properly below (resp., above) 0.5, introduce artificial success (resp., failure) branches into M such that satisfying the original threshold corresponds to having a majority of successful computations. Then, the answer to our entailment problem is *yes* iff the answer of M is *yes* in the majority of its runs. Hardness holds even if we consider PDBs since probabilistic entailment in PDBs (and thus in OpenPDBs) is PP-hard, which correspond to PDBs with empty programs. □

E Proof Sketch of Lemma 10

We have

$$\mathrm{P}(\mathtt{Q},\Sigma) \overset{(1)}{=} \sum_{\substack{\mathcal{D}\models(\mathtt{Q}\cup\Sigma)\\ mods(\mathcal{D},\Sigma)\neq\emptyset}} \mathrm{P}(\mathcal{D}) \overset{(2)}{=} \sum_{\mathcal{D}\models\mathtt{Q}_\Sigma} \mathrm{P}(\mathcal{D}) \overset{(3)}{=} \mathrm{P}(\mathtt{Q}_\Sigma),$$

where (1) follows from Definition 5 and the fact that $\mathrm{P}(\mathcal{D})$ is 0 for all inconsistent worlds $\mathcal{D}$; (2) follows from $\mathtt{Q}_\Sigma$ being the FO-rewriting of $\mathtt{Q}$ w.r.t. Σ; and (3) is the definition of the semantics of $\mathtt{Q}_\Sigma$ in PDBs. □

F Proof Sketch of Theorem 12

By Corollary 11, any polynomial-time algorithm that can evaluate $\mathtt{Q}_\Sigma$ over OpenPDBs also yields the upper and lower probabilities of the OMQ $(\mathtt{Q},\Sigma)$ relative to an OpenPDB, and vice versa. Moreover, by the same result the lower and upper probabilities of *all* rewritings of $\mathtt{Q}$ coincide, and hence the same algorithm can be used for all of them. Thus, if $(\mathtt{Q},\Sigma)$ is unsafe, then $\mathtt{Q}_\Sigma$ must also be unsafe for OpenPDBs. By the dichotomy of (Dalvi and Suciu 2012; Ceylan, Darwiche, and Van den Broeck 2016) and Corollary 11, this implies that evaluating the lower and upper probabilities must be PP-hard, for both $\mathtt{Q}_\Sigma$ and $(\mathtt{Q},\Sigma)$. □

G Proof Sketch of Theorem 14

TGDs with polynomial data complexity relative to PDBs Since we only need to consider a single probability distribution P, we can create multiples of each world (which then correspond to the nondeterministic branches of a Turing machine M), in such a way that the uniform distribution over all thus generated worlds is equivalent to P when each copy is taken to represent its original world. Then, for thresholds properly below (resp., above) 0.5, introduce artificial success (resp., failure) branches into M such that satisfying the original threshold corresponds to having a majority of successful computations. Then, the answer of the probabilistic UCQ entailment problem is *yes* iff the answer of M (regarding entailment of the query under the program) is *yes* in the majority of its runs.

Full, guarded programs relative to OpenPDBs To obtain an upper bound; we can guess a completion (NP) (which is of size polynomial in the size of the input), check its consistency in P, and make a call to a PP oracle to check the probability of the query (as explained before). We answer yes iff the probability exceeds the threshold provided in the original problem.

For the lower bound, we reduce the following $\mathrm{NP}^{\mathrm{PP}}$-complete problem (Wagner 1986), which uses the *counting quantifier* C: decide the validity of the formula

$$\Phi = \exists x_1,\dots,x_\ell\, \mathsf{C}^c\, y_1,\dots,y_m\ \phi,$$

where $\phi = \phi_1 \wedge \dots \wedge \phi_k$ is a propositional formula in CNF, over the variables $x_1,\dots,x_\ell, y_1,\dots,y_m$.

This amounts to checking whether there is a partial assignment for $x_1,\dots,x_\ell$ that admits at least c extensions to $y_1,\dots,y_m$ that satisfy ϕ.

We assume without loss of generality that ϕ contains all clauses of the form $x_j \vee \neg x_j$, $1 \le j \le \ell$, and similarly $y_j \vee \neg y_j$, $1 \le j \le m$; clearly, this does not affect the existence or number of satisfying assignments for ϕ.

We first describe the PDB $\mathcal{P}_\Phi$ that stores the structure of Φ.

- For each variable y_j, $1 \le j \le m$, it contains the tuples $\langle \mathtt{L}(y_j,0) : 0.5\rangle$ and $\langle \mathtt{L}(y_j,1) : 0.5\rangle$, where we view y_j as a constant. These tuple represent the assignments that map y_j to *false* and *true*, respectively.
- For each literal $(\neg)x$ occurring in a clause ϕ_j, $1 \le j \le k$, we add the tuple $\mathtt{D}(x,j,i)$ with probability 1, where $i = 1$, if the literal is positive, and $i = 0$, if the literal is negative.
- We add the tuples $\mathtt{T}(0), \mathtt{S}(0,1), \mathtt{S}(1,2), \dots, \mathtt{S}(k-1,k)$, $\mathtt{K}(k)$, each with probability 1.

Moreover, for each variable x_j, $1 \le j \le \ell$, we need two open tuples $\mathtt{P}(x_j,0)$ and $\mathtt{P}(x_j,1)$ with similar semantics as the $\mathtt{L}$-tuples, and we set $\lambda := 1$. All other tuples over the introduced signature are added to $\mathcal{P}_\Phi$ with probability 0.

We now describe the program Σ. To detect when a clause is satisfied, we use the additional unary predicate $\mathtt{E}$ and the TGDs $\mathtt{P}(x,i), \mathtt{D}(x,j,i) \rightarrow \mathtt{E}(j)$ and $\mathtt{L}(y,i), \mathtt{D}(y,j,i) \rightarrow \mathtt{E}(j)$. However, we still need to ensure that in each world, exactly one of $\mathtt{P}(x,0)$ and $\mathtt{P}(x,1)$ holds, and similarly for $\mathtt{L}$. The clauses $x_j \vee \neg x_j$ and $y_j \vee \neg y_j$ take care of the lower bound; for the variables $x_1,\dots,x_\ell$ we can represent the remaining part of this constraint through the NC $\mathtt{P}(x,0), \mathtt{P}(x,1) \rightarrow \bot$. This ensures that each consistent λ-completion (that satisfies ϕ in an as yet unspecified way) represents exactly one truth assignment for the variables $x_1,\dots,x_\ell$; moreover, every such assignment can be expressed as a consistent λ-completion.

For the variables $y_1,\dots,y_m$, a similar NC would yield only inconsistent completions. Instead, we use the TGDs $\mathtt{L}(y,0), \mathtt{L}(y,1) \rightarrow \mathtt{B}$ and $\mathtt{B}, \mathtt{D}(x,j,i) \rightarrow \mathtt{E}(j)$. These ensure that any inconsistent assignment for $y_1,\dots,y_m$, i.e., one where some y_j is both *true* and *false*, is automatically marked as satisfying the formula, even if the clauses $x_j \vee \neg x_j$ and $y_j \vee \neg y_j$ are not actually satisfied. Since there are exactly $4^m - 3^m$ such assignments (where both $\mathtt{L}(y_j,0)$ and $\mathtt{L}(y_j,1)$ hold for at least one y_j), we can add this number to the probability threshold that we will use in the end. Note that the probability of each individual assignment is 0.25^m since there are $2m$ relevant $\mathtt{L}$-tuples (the other tuples are fixed to 0 or 1 and do not contribute here).

It remains to detect whether *all* clauses of ϕ are satisfied by a consistent assignment, which we do by the means of the TGDs $\mathtt{T}(i), \mathtt{S}(i,j), \mathtt{E}(j) \rightarrow \mathtt{T}(j)$ and $\mathtt{T}(i), \mathtt{K}(i) \rightarrow \mathtt{Z}(i)$ and, finally, the simple CQ $\mathtt{Q} := \exists i\ \mathtt{Z}(i)$. Then it remains to check whether $\overline{\mathrm{P}}_{\mathcal{G}}(\mathtt{Q},\Sigma) > 0.25^m(4^m - 3^m + (c-1))$ holds, where $\mathcal{G} = (\mathcal{P}_\Phi, 1)$ and the program Σ is as described above.

If this is the case, then there is a λ-completion in which the query probability exceeds this value, which means that at least some worlds with non-zero probability entail $(\mathtt{Q},\Sigma)$, i.e., all clauses of ϕ are satisfied. Hence, this λ-completion represents a valid assignment of the variables $x_1,\dots,x_\ell$. Each of the non-zero worlds under this completion represents a unique combination of tuples of the form $\mathtt{L}(y,0)$ and $\mathtt{L}(y,1)$. The worlds where for at least one variable y_j,

$1 \leq j \leq m$, neither $\mathtt{L}(y_j,0)$ nor $\mathtt{L}(y_j,1)$ holds do not satisfy ϕ, and hence do not entail (Q,Σ) and are not counted. Of the remaining worlds, $4^m - 3^m$ automatically entail (Q,Σ). The other worlds represent the actual assignments for $y_1,\dots,y_m$, and hence we know that more than $c-1$ of those satisfy ϕ.

Conversely, if we are given a partial assignment for $x_1,\dots,x_\ell$ that satisfies this property, then it is easy to construct a λ-completion as above and show that it exceeds the given threshold, using the ideas described above.

All TGDs used here are full and guarded. Moreover, only the PDB and the probability threshold depend on the input formula. Hence, the reduction shows $\mathrm{NP}^{\mathrm{PP}}$-hardness of upper probabilistic CQ entailment in GF. □

H Proof Sketch of Theorem 17

Observe that both $\mathcal{P}_0$ and $\mathcal{P}_\cap$ assign the probability 0 to all inconsistent worlds, since $\mathcal{P}_0$ is consistent by assumption, $\mathcal{P}_\cap$ also corresponds to a consistent λ-completion, and consistent λ-completions can assign non-zero probabilities only to consistent worlds. Hence, by Lemma 10, we can reduce the upper and lower query entailment problems in the spirit of Corollary 11 to the same problems for an OpenPDB as follows.

Let $\mathcal{P}$ be the input PDB and $\mathcal{G} = (\mathcal{P},\lambda)$ be the resulting OpenPDB for which we want to compute the probability interval of (Q,Σ) under the intersection semantics. The OpenPDB $\mathcal{G}' = (\mathcal{P}',\lambda)$ is constructed by adding all tuples that do not occur in $\mathcal{P}_\cap$ with probability 0. This construction is polynomial in data complexity: for each ground tuple t, we need to check whether there exist any matching tuples that, together with t, are an instance of the body of some NC in Σ; the number of tuples we have to consider simultaneously is bounded by the length of the longest conjunction in an NC, which is constant. The maximal λ-completion of $\mathcal{G}'$ corresponds to $\mathcal{P}_\cap$, and the minimal one remains $\mathcal{P}_0$. Hence, by Definition 16, Lemma 8 (applied to $\mathcal{G}'$), and Lemma 10, we have $\overline{\mathrm{P}}_{\mathcal{G}}(\mathsf{Q},\Sigma) = \overline{\mathrm{P}}_{\mathcal{G}}(\mathsf{Q}_\Sigma)$ and $\underline{\mathrm{P}}_{\mathcal{G}}(\mathsf{Q},\Sigma) = \underline{\mathrm{P}}_{\mathcal{G}}(\mathsf{Q}_\Sigma)$ for any rewriting Q_Σ of Q relative to Σ.

We can now apply the arguments from the proof of Theorem 12, together with the fact that the construction of $\mathcal{P}'$ from $\mathcal{P}$ is polynomial, to obtain the dichotomy for the intersection semantics. □

I Proof Sketch of Theorem 18

The lower bounds follow from the complexity of UCQ entailment in X, since we can simulate a classical database by a PDB that uses only the probability 1 if we set $\lambda := 0$.

The generic upper bound of $\mathrm{PSPACE}^{\mathbf{C}}$ is obtained as follows: We consider one λ-completion at a time, check its consistency, compute its probability by enumerating all worlds and summing the probabilities of the worlds that entail the query (which can be checked in $\mathbf{C}$), and finally compare the obtained value to p. Since the schema is fixed, the size of each λ-completion and each world is polynomial. Thus, the consistency test (i.e., checking whether the λ-completion does *not* entail $\bot$) is possible in co-$\mathbf{C}$. Moreover, at each step, we have to store only a single λ-completion, a world, and two probability values. Hence, all of this is possible in polynomial space with the help of a $\mathbf{C}$-oracle.

For the case of $\mathbf{C} = \mathrm{NExp}$ and $\lambda = 0$, we do not need to find a consistent λ-completion. It suffices to execute an exponential number of independent entailment tests, each of which is in NExp, and compute the sum of all probabilities as above. In the general case relative to OpenPDBs, we can guess the initial λ-completion in NP, and then use an NExp oracle to both check it for consistency and compute its probability as detailed above. Hence, we obtain an upper bound of $\mathrm{NP}^{\mathrm{NExp}}$, which is equal to P^{NE} by (Hemachandra 1989). □

J Proof Sketch of Theorem 19

Ontology-Mediated Queries relative to PDBs. For the upper bound, we can use a similar approach as in the proof of Theorem 14. However, the entailment test is now NP-complete, and thus for each branch of the PP machine, to check entailment, we make a call to an NP oracle, which yields the $\mathrm{PP}^{\mathrm{NP}}$ upper bound. The proof of $\mathrm{PP}^{\mathrm{NP}}$-hardness can be obtained as a special case of the proof below, where the initial guess of a λ-completion is removed. For this reason, below we show hardness explicitly for $\mathrm{NP}^{\mathrm{PP}^{\mathrm{NP}}}$ instead of $\mathrm{NP}^{\mathrm{PP}}$; however, these two classes coincide (Toda 1989).

Ontology-Mediated Queries relative to OpenPDBs. To obtain the upper bound, we can first guess a λ-completion (NP), and then sum the probabilities of the worlds (PP) that entail the query (NP). Hence, for every guess, we can make a call to a $\mathrm{PP}^{\mathrm{NP}}$ oracle. We can also use this oracle to check consistency of the guessed λ-completion (which is possible in co-NP). This yields the $\mathrm{NP}^{\mathrm{PP}^{\mathrm{NP}}}$ upper bound, which is equal to $\mathrm{NP}^{\mathrm{PP}}$ by (Toda 1989).

It remains to show hardness, for which we reduce the following $\mathrm{NP}^{\mathrm{PP}^{\mathrm{NP}}}$-complete problem (Wagner 1986): decide validity of

$$\Phi = \exists x_1,\dots,x_\ell\, \mathsf{C}^c y_1,\dots,y_m \exists z_1,\dots,z_n\, \phi_1 \wedge \phi_2 \wedge \dots \wedge \phi_k\,,$$

where every ϕ_i is a propositional clause over $x_1,\dots,x_\ell$, $y_1,\dots,y_m$, $z_1,\dots,z_n$, and $k,\ell,m,n \geq 1$. That is, the task is to find an assignment τ to $x_1,\dots,x_\ell$, such that, for at least c of the partial assignments ρ to $x_1,\dots,x_\ell,y_1,\dots,y_m$ that extend τ, the formula $\exists z_1,\dots,z_n\, \rho(\phi_1 \wedge \phi_2 \wedge \dots \wedge \phi_k)$ is true.

As in the proof of Theorem 14, we can assume without loss of generality that ϕ contains all clauses of the form $x_j \vee \neg x_j$, $1 \leq j \leq \ell$, and $y_j \vee \neg y_j$, $1 \leq j \leq m$. We will also assume that each clause ϕ_j contains exactly three literals. This is without loss of generality, since otherwise we can introduce additional existentially quantified variables to abbreviate the clauses, or duplicate literals if the clauses are too short.

The PDB $\mathcal{P}_\Phi$ for the reduction is defined as follows.

- For each variable y_j, $1 \leq j \leq m$, it contains the tuples $\langle \mathtt{L}(y_j,0) : 0.5\rangle$ and $\langle \mathtt{L}(y_j,1) : 0.5\rangle$.
- Each clause ϕ_j is described with the help of a predicate $\mathtt{M}(\cdot,\cdot,\cdot,j)$ of arity 4, which encodes the satisfying

assignments for ϕ_j. For example, consider the clause $\phi_j = x_2 \vee \neg y_4 \vee z_1$. For the satisfying assignment $x_2 \mapsto$ *true*, $y_4 \mapsto$ *true*, $z_1 \mapsto$ *false*, we add the tuple $\mathtt{M}(1,1,0,j)$ with probability 1, and similarly for all other satisfying assignments. There are at most 7 satisfying assignments for each clause.

We again use the open tuples $\mathtt{P}(x_j,0)$ and $\mathtt{P}(x_j,1)$ for the variables x_j, $1 \le j \le \ell$, set $\lambda := 1$, and fix all other possible tuples to the probability 0. We define the program Σ_Φ for the reduction as follows. We again use the NC $\mathtt{P}(x,0), \mathtt{P}(x,1) \rightarrow \bot$ to enforce that the variables x_j, $1 \le j \le \ell$, get a correct truth assignment. However, we do not employ any TGDs. The UCQ for which we will check entailment is

$$\mathsf{Q}_\Phi := (\exists z_1, \dots, z_n\, \psi_1 \wedge \dots \wedge \psi_k) \vee (\exists y\, \mathtt{L}(y,0) \wedge \mathtt{L}(y,1)),$$

where each ψ_j is a conjunction that is derived from ϕ_j depending on the types of the involved variables. We describe the details again on the example clause $\phi_j = x_2 \vee \neg y_4 \vee z_1$. The satisfaction of this clause is encoded by the conjunction $\psi_j = \mathtt{M}(i_1,i_2,z_1,j) \wedge \mathtt{P}(x_2,i_1) \wedge \mathtt{L}(y_4,i_2)$, where i_1, i_2 are additional existentially quantified variables that are local to ψ_j, and j is fixed. Intuitively, ψ_j asserts that the truth assignment for x_2, y_4, and z_1 (given by i_1, i_2, and z_1, respectively) satisfies ϕ_j. The assignment for the variables $x_1, \dots, x_\ell$, $y_1, \dots, y_m$ is fixed by the current λ-completion (using $\mathtt{P}$) and world (using $\mathtt{L}$), respectively, while the assignment for $z_1, \dots, z_n$ is guessed by Q_Φ. Note that the variables $z_1, \dots, z_n$ have to be mapped to 0 or 1, since otherwise they cannot satisfy the $\mathtt{M}$-atoms. An alternative way of satisfying Q_Φ is that $\mathtt{L}$ represents an inconsistent assignment for at least one variable of the form y_j, which again happens in exactly $4^m - 3^m$ worlds. It remains to check whether $\overline{\mathrm{P}}_\mathcal{G}(\mathsf{Q}_\Phi, \Sigma_\Phi) > 0.25^m(4^m - 3^m + (c-1))$ holds relative to the OpenPDB $\mathcal{G} = (\mathcal{P}_\Phi, 1)$, where the program Σ_Φ consists of a single NC $\mathtt{P}(x,0), \mathtt{P}(x,1) \rightarrow \bot$.

If this is the case, then there exists at least one λ-completion that obtains this value. This λ-completion must represent a valid assignment for the variables $x_1, \dots, x_\ell$ since otherwise only $4^m - 3^m$ worlds satisfy $(\mathsf{Q}_\Phi, \Sigma_\Phi)$. Of the 3^m worlds that do not satisfy $\exists y\, \mathtt{L}(y,0) \wedge \mathtt{L}(y,1)$ there are at most 2^m that also satisfy the constraints on the variables $y_1, \dots, y_m$, and hence represent a valid extension to an assignment for $y_1, \dots, y_m$. Of these remaining 2^m worlds, only those satisfy Q_Φ that admit an extension to a truth assignment for $z_1, \dots, z_n$ such that all conjunctions ψ_j, and hence all clauses ϕ_j, are satisfied. Thus, there must be at least c assignments for $y_1, \dots, y_m$ that have such an extension, which means that Φ is valid.

Conversely, if Φ is valid, then there exists an assignment for $x_1, \dots, x_\ell$ (which induces a λ-completion), for which there are at least c extensions to $y_1, \dots, y_m$ (and hence at least $4^m - 3^m + c$ worlds) for which there exists an extension to $z_1, \dots, z_n$ that satisfies all the clauses $\phi_1, \dots, \phi_k$ (and hence $(\mathsf{Q}_\Phi, \Sigma_\Phi)$ is satisfied). This shows that $\overline{\mathrm{P}}_\mathcal{G}(\mathsf{Q}_\Phi, \Sigma_\Phi)$ exceeds the given threshold. Since the reduction is w.r.t. a fixed schema, we did not use any TGDs and the only NC that was used does not depend on Φ, this shows the claim. □