On Constrained Open-World Probabilistic Databases

Tal Friedman and **Guy Van den Broeck**
Department of Computer Science
University of California, Los Angeles
{tal, guyvdb}@cs.ucla.edu

Abstract

Increasing amounts of available data have led to a heightened need for representing large-scale probabilistic knowledge bases. One approach is to use a probabilistic database, a model with strong assumptions that allow for efficiently answering many interesting queries. Recent work on open-world probabilistic databases strengthens the semantics of these probabilistic databases by discarding the assumption that any information not present in the data must be false. While intuitive, these semantics are not sufficiently precise to give reasonable answers to queries. We propose overcoming these issues by using constraints to restrict this open world. We provide an algorithm for one class of queries, and establish a basic hardness result for another. Finally, we propose an efficient and tight approximation for a large class of queries.

1 Introduction

An ubiquitous pursuit in the study of knowledge base representation is the search for a model that can represent uncertainty while simultaneously answering interesting queries efficiently. The key underlying challenge is that these goals are at odds with each other. Modelling *uncertainty* requires additional model complexity. At the same time, the ability to answer *meaningful queries* usually demands restrictive model assumptions. Both of these properties are at odds with the key limiting factor of *tractability*: success in the first two goals is not nearly as impactful if it is not achieved efficiently. Unfortunately, probabilistic reasoning is often computationally hard, even on databases [Roth, 1996; Dalvi and Suciu, 2012].

One approach towards achieving this goal is to begin with a simple model such a probabilistic database (PDB) [Suciu *et al.*, 2011; Van den Broeck and Suciu, 2017]. A PDB models uncertainty, but is inherently simple and makes very strong independence assumptions and closed-world assumptions allowing for tractability on a very large class of queries [Dalvi and Suciu, 2007, 2012]. However, PDBs can fall short under non-ideal circumstances, as their semantics are brittle to incomplete knowledge bases [Ceylan *et al.*, 2016].

To bring PDBs closer to the desired goal, Ceylan *et al.* [2016] propose open-world probabilistic databases (OpenPDB), wherein the semantics of a PDB are strengthened to relax the closed-world assumption. While OpenPDBs maintain a large class of tractable queries, their semantics are so relaxed these queries lose their precision: they model further uncertainty, but in exchange give less useful query answers.

In this work, we aim to overcome these querying challenges, while simultaneously maintaining the degree of uncertainty modeled by OpenPDBs. To achieve this, we propose further strengthening the semantics of OpenPDBs by constraining the mean probability allowed for a relation. These constraints work at the *schematic* level, meaning no additional per-item information is required. They are practically motivated by knowledge of summary statistics, of how many tuples we expect to be true. A theoretical analysis shows that, despite their simplicity, such constraints fundamentally change the difficulty landscape of queries, leading us to propose a general-purpose approximation scheme.

The rest of the paper is organized as follows: Section 2 provides necessary background on relational logic and PDBs, as well as an introduction to OpenPDBs. Section 3 motivates and introduces our construction for constraining OpenPDBs. Section 4 analyses exact solutions subject to these constraints, providing a class of tractable queries along with an algorithm. It also shows that the problem is in general hard, even in some cases where standard PDB queries are tractable. Section 5 investigates an efficient and provably bounded approximation scheme. Section 6 discusses our findings, and summarizes interesting directions that we leave as open problems.

2 Background

This section provides background and motivation for probabilistic databases and their open-world counterparts. Notation and definitions are adapted from Ceylan *et al.* [2016].

2.1 Relational Logic and Databases

We now describe necessary background from *function-free finite-domain* first-order logic. An atom $R(x_1, x_2, ..., x_n)$ consists of a predicate R of arity n, together with n arguments. These arguments can either be *constants* or *variables*. A *ground atom* is an atom that contains no variables. A *formula* is a series of atoms combined with conjunctions ($\wedge$) or disjunctions ($\vee$), and with quantifiers $\forall, \exists$. A *substitution* $\mathsf{Q}[x/t]$ replaces all occurences of x by t in a formula Q.

Scientist
Einstein
Erdős
von Neumann

CoAuthor	
Einstein	Erdős
Erdős	von Neumann

Figure 1: Example relational database. Notice that the first row of the right table corresponds to the atom CoAuthor(Einstein, Erdős).

Scientist	p
Einstein	0.8
Erdős	0.8
von Neumann	0.9
Shakespeare	0.2

CoAuthor		p
Einstein	Erdős	0.8
Erdős	von Neumann	0.9
von Neumann	Einstein	0.5

Figure 2: Example probabilistic database. Tuples are now of the form $\langle t : p \rangle$ where p is the probability of the tuple t being present.

A relational *vocabulary* σ is comprised of a set of predicates $\mathcal{R}$ and a domain $\mathcal{D}$. Using the *Herbrand semantics* [Hinrichs and Genesereth, 2006], the *Herbrand base* of σ is the set of all ground atoms possible given $\mathcal{R}$ and $\mathcal{D}$. A σ-interpretation ω is then an assignment of truth values to every element of the Herbrand base of σ. We say that ω *models* a formula Q whenever ω satisfies Q. This is denoted by $\omega \models \mathsf{Q}$.

A reasonable starting point for the target knowledge base to construct would be to use a traditional *relational database*. Using the standard model-theoretic view [Abiteboul *et al.*, 1995], a relational database for a vocabulary σ is a σ-interpretation ω. Less formally, a relational database consists of a series of relations, each of which corresponds to a predicate. Each relation consists of a series of rows, also called *tuples*, each of which corresponds to an atom of the predicate being true. Any atom not appearing as a row in the relation is considered to be *false*, following the closed-world assumption [Reiter, 1981]. Figure 1 shows an example database.

2.2 Probabilistic Databases

Despite the success of relational databases, their deterministic nature leads to a few shortcomings. A common way to gather a large knowledge base is to apply some sort of statistical model [Carlson *et al.*, 2010; Suchanek *et al.*, 2007; Peters *et al.*, 2014; Dong *et al.*, 2014] which returns a probability value for potential tuples. Adapting the output of such a model to a relational database involves thresholding on the probability value, discarding valuable information along the way. A *probabilistic database* (PDB) circumvents this problem by assigning each tuple a probability.

Definition 1. A *(tuple-independent) probabilistic database* $\mathcal{P}$ for a vocabulary σ is a finite set of tuples of the form $\langle t : p \rangle$ where t is a σ-atom and $p \in [0, 1]$. Furthermore, each t can appear at most once.

Given such a collection of tuples and their probabilities, we are now going to define a *distribution* over relational databases. The semantics of this distribution are given by treating each tuple as an independent random variable.

Definition 2. A probabilistic database $\mathcal{P}$ for vocabulary σ induces a probability distribution over σ-interpretations ω:

$$P_{\mathcal{P}}(\omega) = \prod_{t \in \omega} P_{\mathcal{P}}(t) \prod_{t \notin \omega} (1 - P_{\mathcal{P}}(t))$$

$$\text{where} \quad P_{\mathcal{P}}(t) = \begin{cases} p & \text{if } \langle t : p \rangle \in \mathcal{P} \\ 0 & \text{otherwise} \end{cases}$$

Notice this last statement is again making the closed-world assumption: any tuple that we have no information about is assigned probability zero. Figure 2 shows an example PDB.

Probabilistic Queries In relational databases, the fundamental task we are interested in solving is how to answer queries. The same is true for probabilistic databases, with the only difference being that we are now interested in probabilities over queries. In particular, we are interested in queries that are fully quantified - also known as *Boolean queries*. On a relational database, this corresponds to a query that has an answer of True or False.

For example, on the database given in Figure 1, we might ask if there is a scientist who is a coauthor:

$$\mathsf{Q}_1 = \exists x. \exists y. S(x) \wedge CoA(x, y)$$

If we instead asked this query of the probabilistic database in Figure 2, we would be computing the probability by summing over the worlds in which the query is true:

$$P(\mathsf{Q}_1) = \sum_{\omega \models \mathsf{Q}_1} P_{\mathcal{P}}(\omega)$$

Queries of this form that are a conjunction of atoms are called *conjunctive queries*. They are commonly shortened as:

$$\mathsf{Q}_1 = S(x), CoA(x, y).$$

A disjunction of conjunctive queries is known as a *union of conjunctive queries* (UCQ). UCQs have been shown to live in a dichotomy of efficient evaluation [Dalvi and Suciu, 2012]: computing the probability of a UCQ is either polynomial in the size of the database, or it is $\#P$-hard. This property can be checked through the syntax of a query, and we say that a UCQ is *safe* if it admits efficient evaluation. In the literature of probabilistic databases [Suciu *et al.*, 2011; Dalvi and Suciu, 2012], as well as throughout the rest of this paper, UCQs are the primary query object studied.

Efficient Query Evaluation For probabilistic databases to be useful, we need to be able to effeciently compute the probabilities of queries: we now describe how to do this. Algorithm 1 does this in polynomial time for all queries that can be computed efficiently (known as *safe* queries). We now explain the steps in further detail.

We begin with the assumption that Q has been processed to not contain any constant symbols, and that all variables appear in the same order in repeated predicate occurences in Q. These preprocessing steps are known as *shattering* and *ranking* respectively, and can be done efficiently [Dalvi and Suciu, 2012].

Step 0 covers the base case where Q is simple a tuple, so it looks it up in $\mathcal{P}$. *Step 1* attemps to rewrite the UCQ into a conjunction of UCQs to find decomposable parts. For example, the UCQ $(R(x) \wedge S(y, z)) \vee (S(x, y) \wedge T(x))$ can be

Algorithm 1 $\textbf{Lift}^{\mathbf{R}}(\mathsf{Q}, \mathcal{P})$, abbreviated by $\mathbf{L}(\mathsf{Q})$

Require: UCQ Q , prob. database $\mathcal{P}$ with constants T.
Ensure: The probability $P_{\mathcal{P}}(\mathsf{Q})$

Step 0 *Base of Recursion*
  if Q is a single ground atom t
    if $\langle t : p\rangle \in \mathcal{P}$ **return** p **else return** 0
Step 1 *Rewriting of Query*
  Convert Q to conjunction of UCQ: $\mathsf{Q}_\wedge = \mathsf{Q}_1 \wedge \cdots \wedge \mathsf{Q}_m$
Step 2 *Decomposable Conjunction*
  if $m > 1$ and $\mathsf{Q}_\wedge = \mathsf{Q}_1 \wedge \mathsf{Q}_2$ where $\mathsf{Q}_1 \perp \mathsf{Q}_2$
    return $\mathbf{L}(\mathsf{Q}_1) \cdot \mathbf{L}(\mathsf{Q}_2)$
Step 3 *Inclusion-Exclusion*
  if $m > 1$ but $\mathsf{Q}_\wedge$ has no independent Q_i
    (Do Cancellations First)
    return $\sum_{s\subseteq[m]} (-1)^{|s|+1} \cdot \mathbf{L}\left(\bigvee_{i\in s} \mathsf{Q}_i\right)$
Step 4 *Decomposable Disjunction*
  if $\mathsf{Q} = \mathsf{Q}_1 \vee \mathsf{Q}_2$ where $\mathsf{Q}_1 \perp \mathsf{Q}_2$
    return $1 - (1 - \mathbf{L}(\mathsf{Q}_1)) \cdot (1 - \mathbf{L}(\mathsf{Q}_2))$
Step 5 *Decomposable Existential Quantifier*
  if Q has a *separator variable* x
    return $1 - \prod_{c\in T} (1 - \mathbf{L}(\mathsf{Q}[x/c]))$
Step 6 *Fail* (the query is #P-hard)

written as the conjunction of $(R(x)) \vee (S(x, y) \wedge T(x))$ and $(S(y, z)) \vee (S(x, y) \wedge T(x))$. When multiple conjuncts are found this way, there are two options. If they are symbolically independent (share no symbols, denoted $\perp$), then *Step 2* applies independence and recurse. Otherwise, *Step 3* recurses using the inclusion-exclusion principle, performing cancellations first to maintain efficiency [Dalvi and Suciu, 2012]. If there is only a single UCQ after rewriting, *Step 4* tries to split it into independent parts, applying independence and recursing if anything is found.

Next, *Step 5* searches for a *separator* variable, one which appears in every atom in Q. If x is a separator variable for Q, and a, b are different constants in the domain of x, this means that $\mathsf{Q}[x/a]$ and $\mathsf{Q}[x/b]$ are independent. This independence is again recursively exploited. Finally, if *Step 6* is reached, then the algorithm has failed and the query cannot be computed efficiently [Dalvi and Suciu, 2012].

2.3 Open-World Probabilistic Databases

In the context of automatically constructing a knowledge base, as is done in for example NELL [Carlson *et al.*, 2010] or Google's Knowledge Vault [Dong *et al.*, 2014], making the closed-world assumption is conceptually unreasonable. Conversely, it is also not feasible to include all possible tuples and their probabilities in the knowledge base. The resulting difficulty is that there are an enormous number of probabilistic facts that can be scraped from the internet, and by definition these tools will keep only those with the very highest probability. As a result, knowledge bases like NELL [Carlson *et al.*, 2010], PaleoDeepDive [Peters *et al.*, 2014], and YAGO [Suchanek *et al.*, 2007] consist almost entirely of probabilities above 0.95.

This tells us that the knowledge base we are looking at is fundamentally *incomplete*. In response to this problem, Ceylan *et al.* [2016] propose the notion of a *completion* for a probabilistic database.

Definition 3. A λ-*completion* of a probabilistic database $\mathcal{P}$ is another probabilistic database obtained as follows. For each atom t that does not appear in $\mathcal{P}$, we add tuple $\langle t : p\rangle$ to $\mathcal{P}$ for some $p \in [0, \lambda]$.

Then, we can define the open world of possible databases in terms of the set of distributions induced by all completions.

Definition 4. An *open-world probabilistic database* (OpenPDB) is a pair $\mathcal{G} = (\mathcal{P}, \lambda)$, where $\mathcal{P}$ is a probabilistic database and $\lambda \in [0, 1]$. $\mathcal{G}$ induces a set of probability distributions $K_{\mathcal{G}}$ such that a distribution P belongs to $K_{\mathcal{G}}$ iff P is induced by some λ-completion of probabilistic database $\mathcal{P}$.

Open-World Queries OpenPDBs specify a set of probability distributions rather than a single one, meaning that a given query produces a set of possible probabilities rather than a single one. We focus on computing the minimum and maximum possible probability values that can be achieved by completing the database.

Definition 5. The *probability interval of a Boolean query* Q in OpenPDB $\mathcal{G}$ is $K_{\mathcal{G}}(\mathsf{Q}) = [\underline{P}_{\mathcal{G}}(\mathsf{Q}), \overline{P}_{\mathcal{G}}(\mathsf{Q})]$, where

$$\underline{P}_{\mathcal{G}}(\mathsf{Q}) = \min_{P\in K_{\mathcal{G}}} P(\mathsf{Q}) \qquad \overline{P}_{\mathcal{G}}(\mathsf{Q}) = \max_{P\in K_{\mathcal{G}}} P(\mathsf{Q})$$

In general, computing the probability interval for some first-order Q is not tractable. As observed in Ceylan *et al.* [2016], however, the situation is different for UCQ queries, because they are monotone (they contain no negations). For UCQs, the upper and lower bounds are given respectively by the full completion (where all unknown probabilities are λ), and the closed world database. This is a direct result of the fact that OpenPDBs form a credal set: a closed convex set of probability measures, meaning that probability bounds always come from extreme points [Cozman, 2000].

Furthermore, Ceylan *et al.* [2016] also provide an algorithm for efficiently computing this upper bound corresponding to a full completion, and show that it works whenever the UCQ is safe.

3 Mean-Constrained Completions

This section motivates the need to strengthen the OpenPDB semantics, and introduces our novel probabilistic data model.

3.1 Motivation

The ability to perform efficient query evaluation provides an appealing case for OpenPDBs. They give a more reasonable semantics, better matching their use, and for a large class of queries they come at no extra cost in comparison to traditional PDBs. However, in practice computing an upper bound in this way tends to give results very close to 1. Intuitively, this makes sense: our upper bound comes from simultaneously assuming that *every* possible missing atom has some reasonable probability. While such a bound is easy to compute, it is too strong of a relaxation of the closed-world assumption.

Query	CW	OW	CoOW
$LiLA(x), S(x)$	0	$1 - 10^{-290}$	$1 - 10^{-15}$
$LiSpr(x), S(x)$	0	$1 - 10^{-191}$	0.96

Table 1: Comparison of upper bounds for the same query and database with different model assumptions: Closed-World (CW), Open-World (OW), and Constrained Open-World (CoOW).

Recall the motivation for the initial OpenPDB semantics: statistical knowledge base construction (KBC) tools store only the most likely extracted tuples [Ceylan *et al.*, 2016]. The λ parameter in OpenPDBs is designed to account for this, representing an upper bound on the probability of unobserved tuples. However, this discards other information potentially collected by the KBC system: for example, suppose that a table in our database describes whether or not a person is a scientist. The OpenPDB model will account for the fact that many of the people we discard have a non-zero chance of being a scientist, but it will not take into account the fact that our KBC system observes that fewer than 1% of the population are scientists.

In order to consider a restricted subset of completions representing reasonable situations, we propose directly incorporating these summary statistics. Specifically, we place constraints on the overall probability of a relation across the entire population. In the scientist example, our model only considers completions in which the probability mass of people being scientists totals less than 1%. This allows us to include more information at the domain level, without having more information about each individual.

Example To illustrate the effect this has, consider a schema in which we have 3 relations: $LiLA(x)$ denoting whether one lives in Los Angeles, $LiSpr(x)$ denoting whether one lives in Springfield, and $S(x)$ denoting whether one is a scientist. Using a vocabulary of 500 people where each person is present in at most one relation, Table 1 shows the resulting upper probability bound under different model assumptions, where the constrained open-world restricts at most 50% of mass on $LiLA$, 5% on S, and 0.5% on $LiSpr$. In particular, notice how extreme the difference is in upper bound with and without constraints being imposed. The closed-world probability of both of these queries is always 0, as each person in our database only has a known probability for at most one relation. It is clear that of these three options, the constrained open-world is the most reasonable – the rest of this section formalizes this idea and investigates the resulting properties.

3.2 Formalization

We begin here by defining mean-based constraints, before examining some immediate observations about the structure of the resulting constrained database.

Definition 6. Suppose we have a PDB $\mathcal{P}$, and let $Tup(R) \subseteq \mathcal{P}$ be the set of probabilistic tuples in relation R. Let $\bar{p}$ be a probability threshold. Then a *mean tuple probability constraint* (MTP constraint) φ is a linear constraint of the form

$$\bar{p} \geq \frac{1}{|Tup(R)|} \sum_{\langle t:p \rangle \in Tup(R)} p$$

Definition 7. We say that a λ-completion is φ*-constrained* if the λ-completed database satisfies MTP φ. If it satisfies all of $\Phi = (\varphi_1, \varphi_2, ..., \varphi_n)$, then we say it is Φ-constrained.

Being φ-constrained is not a property of OpenPDBs, but of their PDB completions. Hence, we are interested in the subset of completions that satisfy this property.

Definition 8. An OpenPDB $\mathcal{G} = (\mathcal{P}, \lambda)$ together with MTP constraints Φ induces a set of probability distributions $K_{\mathcal{G}}^{\Phi}$, where distribution P belongs to $K_{\mathcal{G}}^{\Phi}$ iff P is induced by some Φ-constrained λ-completion of $\mathcal{P}$.

Much like with standard OpenPDBs, for a Boolean query Q we are interested in computing bounds on $P(\mathsf{Q})$.

Definition 9. The probability interval of a Boolean query Q in OpenPDB $\mathcal{G}$ with MTP constraints Φ is $K_{\mathcal{G}}^{\Phi}(\mathsf{Q}) = [\underline{P}_{\mathcal{G}}^{\Phi}(\mathsf{Q}), \overline{P}_{\mathcal{G}}^{\Phi}(\mathsf{Q})]$, where

$$\underline{P}_{\mathcal{G}}^{\Phi}(\mathsf{Q}) = \min_{P \in K_{\mathcal{G}}^{\Phi}} P(\mathsf{Q}); \qquad \overline{P}_{\mathcal{G}}^{\Phi}(\mathsf{Q}) = \max_{P \in K_{\mathcal{G}}^{\Phi}} P(\mathsf{Q}).$$

3.3 Completion Properties

A necessary property of OpenPDBs for efficient query evaluation is that they are credal – this is what allows us to consider only a finite subset of possible completions. MTP-constrained OpenPDBs maintain this property.[1]

Proposition 1. *Suppose we have an OpenPDB $\mathcal{G}$ together with MTP constraints Φ. Then the induced set of probability distributions $K_{\mathcal{G}}^{\Phi}$ is credal.*

This property allows us to examine only a finite subset of configurations when looking at potential completions, since query probability bounds of a credal set are always achieved at points of extrema [Cozman, 2000]. Next, we would like to characterize these points of extrema, by showing that the number of tuples not on their own individual boundaries (that is, 0 or λ) is given by the number of MTP constraints.

Theorem 2. *Suppose we have an OpenPDB $\mathcal{G} = (\mathcal{P}, \lambda)$ with MTP constraints Φ, and a UCQ Q. Then there exists a Φ-constrained λ-completion $\mathcal{P}'$ for which $P_{\mathcal{P}'}(\mathsf{Q}) = \overline{P}_{\mathcal{G}}^{\Phi}(\mathsf{Q})$ and that contains some $\mathcal{T} \subseteq \mathcal{P}' \setminus \mathcal{P}$ such that $|\mathcal{T}| \leq |\Phi|$, and*

$$\forall \langle t : p \rangle \in \mathcal{T} : \;\; p \in [0, \lambda], \text{ and}$$
$$\forall \langle t : p \rangle \in (\mathcal{P}' \setminus \mathcal{P}) \setminus \mathcal{T} : \;\; p \in \{0, \lambda\}.$$

That is, our upper bound is given by a completion that has at most $|\Phi|$ added tuples with probability not exactly 0 or λ. Intuitively, each MTP constraint contributes a single non-boundary tuple, which can be thought of as the "leftover" probability mass once the rest has been assigned in full.

This insight allows us to treat MTP query evaluation as a combinatorial optimization problem for the rest of this paper. Thus, we only consider the case where achieving the mean tuple probability exactly leaves us with every individual tuple at its boundary. To see that we can do this, we observe that

[1]Proofs of all theorems and lemmas are available in appendix of the full version of the paper at http://starai.cs.ucla.edu/papers/FriedmanIJCAI19.pdf

Theorem 2 leaves a single tuple per MTP constraint not necessarily on the boundary. But this tuple can always be forced to be on the boundary by very slightly increasing the mean $\bar{p}$ of the constraint, as follows.

Corollary 3. *Suppose we have an OpenPDB $\mathcal{G} = (\mathcal{P}, \lambda)$ with MTP constraints Φ, and a UCQ Q. Suppose further that each relation in $\mathcal{G}$ has at most 1 constraint in Φ, and that each constraint allows adding open-world probability mass exactly divisible by λ. Then there exists a Φ-constrained λ-completion $\mathcal{P}'$, where $K^{\Phi}_{\mathcal{G}}(\mathsf{Q}) = [P_{\mathcal{P}}(\mathsf{Q}), P_{\mathcal{P}'}(\mathsf{Q})]$, and*

$$\forall \langle t : p \rangle \in \mathcal{P}' \setminus \mathcal{P} : \quad p \in \{0, \lambda\}.$$

Our investigation into the algorithmic properties of MTP query evaluation will be focused on constraining a single relation, subject to a single combinatorial budget constraint.

4 Exact MTP Query Evaluation

With Section 3 formalizing MTP constraints and showing that computing upper bounds subject to MTP constraints is a combinatorial problem of choosing which λ-probability tuples to add in the completion, we now investigate exact solutions. With this now being a combinatorial problem, we slightly change our terminology: "adding" an open-world tuple t to a relation means we consider only completions where $P(t) = \lambda$, and a "budget" b for a relation means we can add up to b tuples while still satisfying the MTP constraint.

4.1 An Algorithm for Inversion-Free Queries

We begin by describing a class of queries that admits polytime evaluation subject to an MTP constraint. We first need to define some syntactic properties of queries.

Definition 10. Let Q be a conjunctive query, and let $at(x)$ denote the set of relations containing variable x. We say that Q is *hierarchical* if for any x, y, we have either $at(x) \subseteq at(y)$, $at(y) \subseteq at(x)$, or $at(x) \cap at(y) = \emptyset$.

Intuitively, a conjunctive query being hierarchical indicates that it can either be separated into independent parts (the $at(x) \cap at(y) = \emptyset$ case), or there is some variable that appears in every atom. This simple syntactic property is the basis for determining whether query evaluation on a conjunctive query can be done in polynomial time [Dalvi and Suciu, 2007]. We can further expand on this definition in the context of UCQs.

Definition 11. A UCQ Q is *inversion-free* if each of its disjuncts is hierarchical, and they all share the same hierarchy.[2] If Q is not inversion-free, we say that it has an inversion.

Inversion-free queries represent an especially tractable class of queries for general inference. Since they are hierarchical, they are also safe, meaning query evaluation is efficient. Moreover, they precisely characterize the class of queries that support compilation to a tractable form for performing more complex queries, such as computing any joint distribution [Jha and Suciu, 2011]. This query class remains tractable under MTP constraints.

Theorem 4. *For any inversion-free query Q, evaluating the probability $\overline{P}^{\Phi}_{\mathcal{G}}(\mathsf{Q})$ subject to an MTP constraint is in PTIME.*

[2] See Jha and Suciu [2011] for a more detailed definition.

In order to prove Theorem 4, we provide a polytime algorithm for MTP query evaluation on inversion-free queries. As with OpenPDBs, our algorithm depends on Algorithm 1, the standard lifted inference algorithm for PDBs that was discussed in Section 2.

We now present an algorithm for doing exact MTP query evaluation on inversion-free queries. For brevity, we present the case of a binary relation; the general case follows similarly and can be found in appendix. Suppose that we have a probabilistic database $\mathcal{P}$, a domain T of constants denoted c, a query Q, and an MTP constraint on relation $R(x, y)$ allowing us to add exactly b tuples with probability λ. Suppose that Q immediately reaches Step 5 of Algorithm 1 (other steps will be discussed later), implying that x and y are unique variables in the query. We let $A(c_x, c_y, b)$ denote the upper query probability of $\mathsf{Q}(x/c_x, y/c_y)$ subject to an MTP constraint allowing budget b on R restricted to $x = c_x, y = c_y$. That is, A tells us the highest probability we can achieve for a partial assignment given a fixed budget. Observe that we can compute all entries of A using a slight modification of Algorithm 1 where we compute probabilities with and without each added tuple. This will take time polynomial in $|T|$.

Next, we impose an ordering $c_1, \ldots, c_{|T|}$ on the domain. Then we let $D(j, c_y, b)$ denote the upper query probability of

$$\bigvee_{c \in \{c_1, \ldots c_j\}} \mathsf{Q}(x/c, y/c_y)$$

with a budget of b on the relevant portions of R. Then $D(|T|, c_y, b)$ considers all possible substitutions in our first index, meaning we have effectively removed a variable. Doing this repeatedly would allow us to perform exact MTP query evaluation. However, D is non-trivial to compute, and cannot be done by simply modifying Algorithm 1. Instead, we observe the following recurrence:

$$\begin{aligned} D(j+1, y/c_y, b) = \\ \max_{k \in \{1, \ldots, b\}} 1 - (1 - D(j, y/c_y, b-k)) \\ \cdot (1 - A(x/c_{j+1}, y/c_y, k)) \end{aligned}$$

Intuitively, this recurrence says that since the tuples from each fixed constant are all independent, we do not need to store which budget configuration on the first j constants got us our optimal solution. Thus, when we add the $j+1$th constant, we just need to check each possible value we could assign to our new constant, and see which gives the overall highest probability. This recurrence can be implemented efficiently, yielding a dynamic programming algorithm that runs in time polynomial in the domain size and budget.

Finally, we would like to generalize this algorithm beyond the assumption that Q immediately reaches Step 5 of Algorithm 1. Looking at other cases, we see that Steps 0 and 1 have no effect on this recurrence, and that Steps 2 and 4 correspond to multiplicative factors. For a query that reaches Step 3 (inclusion-exclusion), we need to construct such A and D for each term in the inclusion-exclusion sum, and follow the analogous recurrence.

Notice that the modified algorithm would only work in the case where we can always pick a common variable for all

sub-queries to do dynamic programming on – that is, when the query is inversion-free, as was our assumption. If the sub-calls generated by inclusion-exclusion do not share a common variable hierarchy, and thus an order for using our dynamic programming algorithm, we suffer an exponential blowup.

4.2 Queries with Inversion

We now show that allowing for inversions in safe queries can cause MTP query evaluation to become NP-hard. Interestingly, this means that MTP constraints fundamentally change the difficulty landscape of query evaluation.

To show this, we investigate the following UCQ query.

$$M_0 = \exists x \exists y \exists z \, (R(x,y,z) \wedge U(x)) \vee (R(x,y,z) \wedge V(y)) \\ \vee (R(x,y,z) \wedge W(z)) \vee (U(x) \wedge V(y)) \\ \vee (U(x) \wedge W(z)) \vee (V(y) \wedge W(z))$$

A key observation here is that the query M_0 is a *safe* UCQ. That is, if we ignore constraints and evaluate it subject to the closed- or open-world semantics, computing the probability of the query would be polynomial in the size of the database. We now show that this is *not* the case for open-world query evaluation subject to a single MTP constraint on R.

Theorem 5. *Evaluating the upper query probability bound $\overline{P}^{\Phi}_{\mathcal{G}}(M_0)$ subject to an MTP constraint Φ on R is NP-hard.*

The full proof of Theorem 5 can be found in appendix, showing a reduction from the NP-complete 3-dimensional matching problem to computing $\overline{P}^{\Phi}_{\mathcal{G}}(M_0)$ with an MTP constraint on R. It uses the following intuitive correspondence.

Definition 12. Let X, Y, Z be finite disjoint sets representing nodes, and let $T \subseteq X \times Y \times Z$ be the set of available hyperedges. Then $M \subseteq T$ is a *matching* if for any distinct triples $(x_1, y_1, z_1) \in M, (x_2, y_2, z_2) \in M$, we have that $x_1 \neq x_2, y_1 \neq y_2, z_1 \neq z_2$. The *3-dimensional matching* decision problem is to determine for a given X, Y, Z, T and positive integer k if there exists a matching M with $|M| \geq k$.

The set of available tuples for R will correspond to all edges in T. The MTP constraint on R forces a decision on which subset of T to add to the database.

However, if we simply queried to maximize $P(R(x,y,z))$, this completion need not correspond to a matching. Instead, we have the disjunct $R(x,y,z) \wedge U(x)$ which is maximized when each tuple chosen from R has a different x value. Similar disjuncts for y and z ensure that the query is maximized when using distinct y and z values. Putting all of these together ensures that the query probability is maximized when the subset of tuples chosen to complete R form a matching.

Finally, the last part of the query $(U(x) \wedge V(y)) \vee (U(x) \wedge W(z)) \vee (V(y) \wedge W(z))$ ensures that inference on M_0 is tractable, but it is unaffected by the choice of tuples in R.

5 Approximate MTP Query Evaluation

With Section 4.2 answering definitively that a general-purpose algorithm for evaluating MTP query bounds is unlikely to exist, even when restricted to safe queries, an approximation is the logical next step. We now restrict our discussion to situations where we constrain a single relation, and dig deeper into the properties of MTP constraints to show their submodular structure. We then exploit this property to achieve efficient bounds with guarantees.

5.1 On the Submodularity of Adding Tuples

To formally define and prove the submodular structure of the problem, we analyze query evaluation as a set function on adding tuples. We begin with a few relevant definitions.

Definition 13. Suppose that we have an OpenPDB $\mathcal{G}$, with an MTP constraint φ on a single relation R, and we let $\mathbf{O}$ be the set of possible tuples we can add to R. Then the *set query probability* function $S_{\mathcal{P},\mathsf{Q}} : 2^{\mathbf{O}} \to [0,1]$ is defined as

$$S_{\mathcal{P},\mathsf{Q}}(X) = P_{\mathcal{P} \cup \{\langle t:\lambda \rangle | t \in X\}}(\mathsf{Q}).$$

Intuitively, this function describes the probability of the query as a function of which open tuples have been added. It provides a way to reason about the combinatorial properties of this optimization problem. Observe that $S_{\mathcal{P},\mathsf{Q}}(\emptyset)$ is the closed-world probability of the query, while $S_{\mathcal{P},\mathsf{Q}}(\mathbf{O})$ is the open-world probability.

We want to show that $S_{\mathcal{P},\mathsf{Q}}$ is a submodular set function.

Definition 14. A *submodular set function* is a function $f : 2^{\Omega} \to \mathbb{R}$ such that for every $X \subseteq Y \subseteq \Omega$, and every $x \in \Omega \setminus Y$, we have that

$$f(X \cup \{x\}) - f(X) \geq f(Y \cup \{x\}) - f(Y).$$

Theorem 6. *The set query probability function $S_{\mathcal{P},\mathsf{Q}}$ is submodular for any tuple independent probabilistic database $\mathcal{P}$ and UCQ query Q without self-joins.*

This gives us the desired submodularity property, which we can exploit to build efficient approximation algorithms.

5.2 From Submodularity to Approximation

Given the knowledge that the probability of a safe query without self-joins is submodular in the completion of a single relation, we are now tasked with using this to construct an efficient approximation. Since we further know the probability is also monotone as we have restricted our language to UCQs, Nemhauser *et al.* [1978] tell us that we can get a $1 - \frac{1}{e}$ approximation using a simple greedy algorithm. The final requirement to achieve the approximation described in Nemhauser *et al.* [1978] is that our set function must have the property that $f(\emptyset) = 0$. This can be achieved in a straightforward manner as follows.

Definition 15. In the context of the set query probability function of Definition 13, the *normalized* set query probability function $S'_{\mathcal{P},\mathsf{Q}} : 2^{\mathbf{O}} \to [0,1]$ is defined as

$$S'_{\mathcal{P},\mathsf{Q}}(X) = P_{\mathcal{P} \cup \{\langle t:\lambda \rangle | t \in X\}}(\mathsf{Q}) - P_{\mathcal{P}}(\mathsf{Q}).$$

Proposition 7. *Any normalized set query probability function $S'_{\mathcal{P},\mathsf{Q}}$ is monotone, submodular, and satifies $S'_{\mathcal{P},\mathsf{Q}}(\emptyset) = 0$.*

By simply normalizing the set query probability function, we can now directly apply the greedy approximation described in Nemhauser *et al.* [1978]. We slightly modify Algorithm 1 to efficiently compute the next best tuple to add based on the current database, and add it. This is repeated

until adding another tuple would violate the MTP constraint. Finally, we say that $P_{Greedy}(\mathsf{Q})$ is the approximation given by this greedy algorithm and recall that the true upper bound is $\overline{P}^{\Phi}_{\mathcal{G}}(\mathsf{Q})$. We observe that $P_{Greedy}(\mathsf{Q}) \leq \overline{P}^{\Phi}_{\mathcal{G}}(\mathsf{Q})$. Furthermore, Nemhauser *et al.* [1978] tells us the following:

$$P_{Greedy}(\mathsf{Q}) - P_{\mathcal{P}}(\mathsf{Q}) \geq (1 - \frac{1}{e})(\overline{P}^{\Phi}_{\mathcal{G}}(\mathsf{Q}) - P_{\mathcal{P}}(\mathsf{Q}))$$

Combining these and multiplying through gives us the following upper and lower bound on the desired probability.

$$P_{Greedy}(\mathsf{Q}) \leq \overline{P}^{\Phi}_{\mathcal{G}}(\mathsf{Q}) \leq \frac{e \cdot P_{Greedy}(\mathsf{Q}) - P_{\mathcal{P}}(\mathsf{Q})}{e - 1}$$

It should be noted that depending on the query and database, it is possible for this upper bound to exceed 1.

6 Discussion, Future & Related Work

We propose the novel problem of constraining open-world probabilistic databases at the schema level, without having any additional ground information over individuals. We introduced a formal mechanism for doing this, by limiting the *mean tuple probability* allowed in any given completion, and then sought to compute bounds subject to these constraints. We now discuss remaining open problems and related work.

Section 4 showed that there exists a query that is NP-hard to compute exactly, and also presented a tractable algorithm for a class of inversion-free queries. The question remains how hard the other queries are – in particular, is the algorithm presented complete. Is there a complexity dichotomy, that is, a set of syntactic properties that determine the hardness of a query subject to MTP constraints. Questions of this form are a central object of study in probabilistic databases. It has been explored for conjunctive queries [Dalvi and Suciu, 2007], UCQs [Dalvi and Suciu, 2012], and a more general class of queries with negation [Fink and Olteanu, 2016].

The central goal of our work is to find stronger semantics based on OpenPDBs, while still maintaining their desirable tractability. This notion of achieving a powerful semantics while maintaining tractability is a common topic of study. De Raedt and Kimmig [2015] study this problem by using a probabilistic interpretation of logic programs to define a model, leading to powerful semantics but a more limited scope of tractability [Fierens *et al.*, 2015]. Description logics [Nardi *et al.*, 2003] are a knowledge representation formalism that can be used as the basis for a semantics. This is implemented in a probabilistic setting in, for example, probabilistic ontologies [Riguzzi *et al.*, 2012, 2015], probabilistic description logics [Heinsohn, 1994], probabilistic description logic programs [Lukasiewicz, 2005], or the bayesian description logics [Ceylan and Peñaloza, 2014].

Probabilistic databases in particular are of interest due to their simplicity and practicality. Foundational work defines a few types of probabilistic semantics, and provides efficient algorithms as well as when they can be applied [Dalvi and Suciu, 2004, 2007, 2012]. These algorithms along with practical improvements are implemented as industrial level systems such as MystiQ [Ré and Suciu, 2008], SPROUT [Olteanu *et al.*, 2009], MayBMS [Huang *et al.*, 2009], and Trio which implements the closely related Uncertainty-Lineage Databases [Benjelloun *et al.*, 2007].

Problems outside of simple query evaluation are also points of interest for PDBs, for example the most probable database problem [Gribkoff *et al.*, 2014], or the problem of ranking the top-k results [Ré *et al.*, 2007]. In the context of OpenPDBs in particular, Grohe and Lindner [2018] study the notion of an infinite open world, using techniques from analysis to explore when this is feasible. Borgwardt *et al.* [2017] study an orthogonal way to introduce constraints on OpenPDBs to make the probability bounds realistic, by adding logical constraints based on an ontology.

Finally, probabilistic databases are closely related to other statistical relational models such as Markov logic networks [Richardson and Domingos, 2006] and probabilistic soft logic [Kimmig *et al.*, 2012]. These models implicitly support the open-world assumption, although inference will not be efficient in general given the large number of random variables induced by the open world.

Acknowledgements

The authors thank Yitao Liang and YooJung Choi for helpful feedback. This work is partially supported by NSF grants #IIS1657613, #IIS-1633857, #CCF-1837129, DARPA XAI grant #N66001-17-2-4032, NEC Research and gifts from Intel and Facebook Research.

References

Serge Abiteboul, Richard Hull, and Victor Vianu. Foundations of databases. 1995.

Omar Benjelloun, Anish Das Sarma, Alon Y. Halevy, Martin Theobald, and Jennifer Widom. Databases with uncertainty and lineage. *The VLDB Journal*, 17:243–264, 2007.

Stefan Borgwardt, Ismail Ilkan Ceylan, and Thomas Lukasiewicz. Ontology-mediated queries for probabilistic databases. In *AAAI*, 2017.

Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka, and Tom M. Mitchell. Toward an architecture for never-ending language learning. In *AAAI*, 2010.

İsmail İlkan Ceylan and Rafael Peñaloza. The bayesian description logic $\mathcal{BEL}$. In Stéphane Demri, Deepak Kapur, and Christoph Weidenbach, editors, *Automated Reasoning*, pages 480–494, Cham, 2014. Springer International Publishing.

Ismail Ilkan Ceylan, Adnan Darwiche, and Guy Van den Broeck. Open-world probabilistic databases. In *KR*, 2016.

Fábio Gagliardi Cozman. Credal networks. *Artif. Intell.*, 120:199–233, 2000.

Nilesh N. Dalvi and Dan Suciu. Efficient query evaluation on probabilistic databases. *The VLDB Journal*, 16:523–544, 2004.

Nilesh N. Dalvi and Dan Suciu. The dichotomy of conjunctive queries on probabilistic structures. In *PODS*, 2007.

Nilesh N. Dalvi and Dan Suciu. The dichotomy of probabilistic inference for unions of conjunctive queries. *J. ACM*, 59:30:1–30:87, 2012.

Luc De Raedt and Angelika Kimmig. Probabilistic (logic) programming concepts. *Machine Learning*, 100:5–47, 2015.

Xin Luna Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*, pages 601–610, 2014. Evgeniy Gabrilovich Wilko Horn Ni Lao Kevin Murphy Thomas Strohmann Shaohua Sun Wei Zhang Geremy Heitz.

Daan Fierens, Guy Van den Broeck, Joris Renkens, Dimitar Sht. Shterionov, Bernd Gutmann, Ingo Thon, Gerda Janssens, and Luc De Raedt. Inference and learning in probabilistic logic programs using weighted boolean formulas. *TPLP*, 15:358–401, 2015.

Robert Fink and Dan Olteanu. Dichotomies for queries with negation in probabilistic databases. *ACM Trans. Database Syst.*, 41:4:1–4:47, 2016.

Eric Gribkoff, Guy Van den Broeck, and Dan Suciu. The most probable database problem. *Proc. BUDA*, 2014.

Martin Grohe and Peter Lindner. Probabilistic databases with an infinite open-world assumption. *CoRR*, abs/1807.00607, 2018.

Jochen Heinsohn. Probabilistic description logics. In *UAI*, 1994.

Timothy Hinrichs and Michael Genesereth. Herbrand logic. Technical Report LG-2006-02, Stanford University, 2006.

Jiewen Huang, Lyublena Antova, Christoph Koch, and Dan Olteanu. Maybms: a probabilistic database management system. In *SIGMOD Conference*, 2009.

Abhay Jha and Dan Suciu. Knowledge compilation meets database theory: Compiling queries to decision diagrams. *Theory of Computing Systems*, 52:403–440, 2011.

Angelika Kimmig, Stephen H. Bach, Matthias Broecheler, Bert Huang, and Lise Getoor. A short introduction to probabilistic soft logic. In *NIPS 2012*, 2012.

Thomas Lukasiewicz. Probabilistic description logic programs. In *ECSQARU*, 2005.

D. Nardi, Werner Nutt, and Francesco M. Donini. The description logic handbook: Theory, implementation, and applications. In *Description Logic Handbook*, 2003.

G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functions—i. *Mathematical Programming*, 14(1):265–294, Dec 1978.

Dan Olteanu, Jiewen Huang, and Christoph Koch. Sprout: Lazy vs. eager query plans for tuple-independent probabilistic databases. *2009 IEEE 25th International Conference on Data Engineering*, pages 640–651, 2009.

Shanan E Peters, Ce Zhang, Miron Livny, and Christopher Ré. A machine reading system for assembling synthetic paleontological databases. In *PloS one*, 2014.

Christopher Ré and Dan Suciu. Managing probabilistic data with mystiq: The can-do, the could-do, and the can't-do. In *SUM*, 2008.

Christopher Ré, Nilesh N. Dalvi, and Dan Suciu. Efficient top-k query evaluation on probabilistic data. *2007 IEEE 23rd International Conference on Data Engineering*, pages 886–895, 2007.

Raymond Reiter. On closed world data bases. In *Readings in artificial intelligence*, pages 119–140. Elsevier, 1981.

Matthew Richardson and Pedro M. Domingos. Markov logic networks. *Machine Learning*, 62:107–136, 2006.

Fabrizio Riguzzi, Elena Bellodi, Evelina Lamma, and Riccardo Zese. Epistemic and statistical probabilistic ontologies. In *URSW*, 2012.

Fabrizio Riguzzi, Elena Bellodi, Evelina Lamma, and Riccardo Zese. Reasoning with probabilistic ontologies. In *IJCAI*, 2015.

Dan Roth. On the hardness of approximate reasoning. *Artificial Intelligence*, 82(1-2):273–302, 1996.

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: A Core of Semantic Knowledge. In *16th International Conference on the World Wide Web*, pages 697–706, 2007.

Dan Suciu, Dan Olteanu, R. Christopher, and Christoph Koch. *Probabilistic Databases*. Morgan & Claypool Publishers, 1st edition, 2011.

Guy Van den Broeck and Dan Suciu. *Query Processing on Probabilistic Data: A Survey*. Foundations and Trends in Databases. Now Publishers, August 2017.

A Proofs of Theorems, Lemmas, and Propositions

A.1 Proof of Proposition 1

Proof. To prove this, we need to show that $K^{\Phi}_{\mathcal{G}}$ is both closed and convex.

Due to the way our constraints are defined, we know that $K^{\Phi}_{\mathcal{G}} = K_{\mathcal{G}} \cap K^{\Phi}$, where K^{Φ} is the set of all completions satisfying Φ (but not necessarily having all tuple probabilities $\leq \lambda$). We already know that $K_{\mathcal{G}}$ is credal, and thus closed and convex. K^{Φ} is a half-space, which we also know is closed and convex. The intersection of closed spaces is closed, and the intersection of convex spaces is convex, so $K^{\Phi}_{\mathcal{G}}$ is credal. □

A.2 Proof of Theorem 2

Lemma 8. *If $S \subseteq \mathbb{R}^n$ is a set formed by the intersection of $k < n$ half-spaces, S has no points of extrema.*

Proof. Written as a set of linear equalities, the solution clearly must have at least 1 degree of freedom. This indicates that for any potential extrema point x, one can move in either direction along this degree of freedom to construct an open line intersecting x, but entirely contained in S. □

Proof. Since $K^{\Phi}_{\mathcal{G}}$ is credal, we are interested here in determining the point of extrema of $K^{\Phi}_{\mathcal{G}}$, as this will tell us precisely which completions can represent boundaries.

Consider the construction of the set $K^{\Phi}_{\mathcal{G}}$, and suppose that there are d possible open-world tuples, meaning that $K^{\Phi}_{\mathcal{G}} \subseteq \mathbb{R}^d$. As we observed in the proof of Theorem 1, $K^{\Phi}_{\mathcal{G}} = K_{\mathcal{G}} \cap K^{\Phi}$, where K^{Φ} is the set of all completions satisfying Φ. We now make three key observations about these sets:

1. Each individual possible open-world tuple is described by the intersection of 2 half-spaces: that is, the tuple on dimension i is described by $x_i \geq 0$ and $x_i \leq \lambda$. $K_{\mathcal{G}}$ is the intersection of all $2d$ of these half-spaces.
2. For any individual open-world tuple, the boundaries of the two half-spaces that describe it cannot intersect each other.
3. An MTP constraint is a linear constraint, and thus can be described by a single half-space. So K^{Φ} is described by the intersection of these $|\Phi|$ half-spaces.

Observations 1 and 3, together with Lemma 8 tells us that any point of extrema of $K^{\Phi}_{\mathcal{G}}$ must be given by the intersection of the boundaries of at least d of the half-spaces that form $K^{\Phi}_{\mathcal{G}}$. Observation 3 tells us that at most $|\Phi|$ of these half-spaces come from MTP constraints, leaving the boundaries of at least $d - |\Phi|$ half-spaces which come from $K_{\mathcal{G}}$. Finally, observation 2 tells us that each of these $d - |\Phi|$ half-spaces is describing a different open world tuple. But this means we must have at least $d - |\Phi|$ tuples which lie on the boundary of one of their defining half-spaces: they must be either 0 or λ. □

A.3 Proof of Theorem 5

Before we present the formal proof, we state and prove 2 Lemmas we will need.

Lemma 9. *Suppose we have two completions P_1 and P_2 of R, which only differ on a single triple, that is $P_1 = P_0 \cup \{x_1, y_1, z_1\}$ and $P_2 = P_0 \cup \{x_2, y_2, z_2\}$. Further suppose that $y_1 = y_2$, $z_1 = z_2$, and that P_0 contains no triples with x-value x_1, but does contain at least 1 triple with x-value x_2. Then $P_1(M_0) > P_2(M_0)$.*

Proof. Let us directly examine Δ, the logical formula found by grounding M_0. Since M_0 is a union of conjunctive queries, Δ must be a DNF. Each disjunct either does not contain R, in which case it does not vary with the choice of completion, or it contains it exactly once. Any disjunct containing an atom of R not assigned probability by a completion is logically false.

In order to prove that $P_1(M_0) > P_2(M_0)$, let us compare the ground atoms that result from each. It is clear that the only spot on which they differ is on disjuncts involving $R(x_1, y_1, z_1)$ or $R(x_2, y_1, z_1)$. Any disjuncts involving one of these and V or W will also have an identical effect on the probability of the query, since the completions are identical over y and z.

Finally this means we need to compare the term $R(x_1, y_1, z_1), U(x_1)$ with the term $R(x_2, y_1, z_1), U(x_2)$. Observe that we know P_0 contains triples with x-value x_2, which means the term only contributes new probability mass when $U(x_2)$ is true *and* none of the other triples involving x_2 are true. However, P_0 does not contain any triples with x-value x_1, so the term $R(x_1, y_1, z_1), U(x_1)$ contributes the maximum probability possible. Thus, for any choice of probabilities on U such that $U(x_2) < 1$, we have that $P_1(M_0) > P_2(M_0)$. □

Lemma 10. *The upper bound $\overline{P}(M_0)$ subject to an MTP constraint on R allowing for $k \cdot \lambda$ total probability mass is maximized if and only if P is a completion formed by a matching of size k, where k is the maximum number of tuples with probability λ that can be added to R in M_0.*

Proof. Observe that if we begin with a completion given by a matching, we can repeatedly apply Lemma 9 to arrive at any completion. Thus a completion given by a matching must have higher probability than any completion not given by a matching. □

Finally, we are ready to present the proof of Theorem 5.

Proof. Suppose we are given an instance of a 3-dimensional matching problem X, Y, Z, T and an integer k. Let $U(x), V(y), W(z)$ be 0.8 wherever $x \in X, y \in Y$, or $z \in Z$ respectively, and 0 everywhere else. Additionally, let $R(x, y, z)$ be unknown for any $(x, y, z) \in T$, and 0 otherwise. Finally, we place an MTP constraint on R ensuring that at most k tuples can be added, and let $\lambda = 0.8$. Then Lemma 10 tells us that M_0 evaluated on this database will be maximized if and only if the completion used corresponds to a matching of size k. We determine this probability P_{max} using a standard probabilistic database query algorithm, and

fixing R to have entries 0.8 for some disjoint set of triples. Observe that this computation is easy if we allow any tuples from R to have non-zero probability: this corresponds to a matching in the setting where all hyperedges are available, but will still give the same query probability.

Finally, we use our oracle for MTP constrained query evaluation to check $\overline{P}(M_0)$ with the database we constructed from the matching problem. We compare the upper bound given by the oracle, and if it is equal to P_{max}, Lemma 10 tells us that a matching of size k does exist. Similarly, if the upper bound given by the oracle is lower than P_{max}, Lemma 10 tells us a matching of size k does not exist. □

A.4 Proof of Theorem 6

Proof. Without directly computing probabilities, let us inspect Δ, the logical formula we get by grounding Q. Q is a union of conjunctive queries, and thus Δ is a DNF. Each disjunct can contain our constrained relation R at most once due to the query not having self-joins, and any one of these disjuncts containing an atom of R not assigned any probability is logically false.

Next, to show that $S_{\mathcal{P},\mathsf{Q}}$ is submodular, let $X \subseteq Y \subseteq \mathbf{O}$, and let $x \in \mathbf{O} \setminus Y$ be given. We assign names to the following subformulas of Δ

- α (β) is the disjunction of all disjuncts of Δ which are not logically false due to missing R_B tuples in X (Y)
- γ is the disjunction of all disjuncts of Δ containing the tuple x

Additionally, since $X \subseteq Y$, we also know that $\alpha \Rightarrow \beta$. Now, we make a few observations relating these quantities with our desired values for submodularity:

- $S_{\mathcal{P},\mathsf{Q}}(X) = P(\alpha)$
- $S_{\mathcal{P},\mathsf{Q}}(Y) = P(\beta)$
- $S_{\mathcal{P},\mathsf{Q}}(X \cup \{x\}) = P(\alpha \vee \gamma)$
- $S_{\mathcal{P},\mathsf{Q}}(Y \cup \{x\}) = P(\beta \vee \gamma)$

Finally, we have the following:

$$\begin{aligned} S_{\mathcal{P},\mathsf{Q}}(X \cup \{x\}) - S_{\mathcal{P},\mathsf{Q}}(X) &= P(\alpha \vee \gamma) - P(\alpha) \\ &= P(\neg\alpha \wedge \gamma) \\ &\geq P(\neg\beta \wedge \gamma) \\ &= P(\beta \vee \gamma) - P(\beta) \\ &= S_{\mathcal{P},\mathsf{Q}}(Y \cup \{x\}) - S_{\mathcal{P},\mathsf{Q}}(Y) \end{aligned}$$

□

B General Algorithm for Inversion-Free Queries

We now present an algorithm for doing exact MTP query evaluation on inversion-free queries. Suppose that we have a probabilistic database $\mathcal{P}$, a domain T of constants denoted c, a query Q, and an MTP constraint on relation $R(x_1, x_2, \ldots, x_r)$ allowing us to add B tuples. For any $I \subseteq \{1, \ldots, r\}$, we let $A(x_{i_1}/c_{i_1}, x_{i_2}/c_{i_2}, \ldots, x_{i_{|I|}}/c_{i_{|I|}}, b)$ denote the upper query probability of $\mathsf{Q}(x_{i_1}/c_{i_1}, x_{i_2}/c_{i_2}, \ldots, x_{i_{|I|}}/c_{i_{|I|}})$ subject to an MTP constraint allowing budget b on the relevant portion of R. That is, A tells us the highest probability we can achieve for a partial assignment given a fixed budget. Observe that we can compute all entries of A using a slight modification of Algorithm 1. This will take time polynomial in $|T|$.

Next, we impose an ordering $c_1, c_2, \ldots, c_{|T|}$ on the domain. For any $I \subseteq \{1, \ldots, r\}$, we let $D(j, x_{i_2}/c_{i_2}, \ldots, x_{i_{|I|}}/c_{i_{|I|}}, b)$ denote the upper query probability of

$$\bigvee_{c \in \{c_1, \ldots c_j\}} \mathsf{Q}(x_{i_1}/c, x_{i_2}/c_{i_2}, \ldots, x_{i_{|I|}}/c_{i_{|I|}}) \tag{1}$$

with a budget of b on the relevant portions of R. Then $D(|T|, x_{i_2}/c_{i_2}, \ldots, x_{i_{|I|}}/c_{i_{|I|}}, b)$ considers all possible substitutions in our first index, meaning we no longer need to worry about it. Doing this repeatedly would allow us to perform exact MTP query evaluation. However, D is non-trivial to compute, and cannot be done by simply modifying Algorithm 1. Instead, we observe the following recurrence:

$$\begin{aligned} &D(j+1, x_{i_2}/c_{i_2}, \ldots, x_{i_{|I|}}/c_{i_{|I|}}, b) \\ &\quad = \max_{k \in \{1,\ldots,b\}} 1 - D(j, x_{i_2}/c_{i_2}, \ldots, x_{i_{|I|}}/c_{i_{|I|}}, b-k) \\ &\quad \cdot A(x_{i_1}/c_{j+1}, x_{i_2}/c_{i_2}, \ldots, x_{i_{|I|}}/c_{i_{|I|}}, k) \end{aligned}$$

Intuitively, this recurrence is saying that since the tuples from each fixed constant are independent of each other, we can add a new constant to our vocabulary by simply considering all combinations of budget assignments. This recurrence can be implemented efficiently, yielding a dynamic programming algorithm that runs in time polynomial in the domain size and budget.

The keen reader will now observe that the above definition and recurrence only make sense if Q immediately reaches Step 5 of Algorithm 1. While this is true, we see that Steps 0 and 1 have no effect on this recurrence, and Steps 2 and 4 correspond to multiplicative factors. For a query that reaches Step 3: inclusion-exclusion, we indeed need to construct such matrices for each sub-query. Notice that the modified algorithm would only work in the case where we can always pick a common x_i for all sub-queries to do dynamic programming on - that is, when the query is inversion-free.