Query Containment of Tier-2 Queries over a Probabilistic Database *

Katherine F. Moore, Vibhor Rastogi, Christopher Ré, Dan Suciu
kfm, vibhor, chrisre, suciu@cs.washington.edu

Department of Computer Science & Engineering
The University of Washington, Seattle

Abstract. We study the containment problem for a query language over probabilistic relational databases that allows queries like *"is the probability that* q_1 *holds greater than* 0.2 *and the probability that* q_2 *holds greater than* 0.6*?"* where q_1 and q_2 are Boolean conjunctive queries. In addition to being a fundamental problem in its own right, the containment problem is the key problem that an optimizer must solve for many standard optimizations (such as picking up an index or using a materialized view). Our main technical result is that the containment problem is decidable, and we give an EXPSPACE-algorithm based on linear programming for it. We believe that we are the first to study the containment problem for any such probabilistic languages.

1 Introduction

An increasing number of data-centric applications are forced to cope with imprecision: in RFID applications, it is difficult to precisely determine the exact location of people or objects [5, 14], in data integration, researchers have allowed schema mappings to be imprecise to lower the overall cost of integration [6, 8], in information extraction, the state-of-the-art techniques produce extractions automatically, but with a loss of precision [12, 20]. Motivated by these applications, researchers have created systems that model the imprecision in the data using probability theory [2, 13, 16, 17].
The first generation of these systems (such as Trio [19], Mystiq [4]) allow a user to pose a standard query (say in SQL). It is then the job of the database system to combine the probabilities in the data to produce a set of result tuples, where each tuple is annotated with a probability score that reflects the extent to which that tuple is an answer to the posed query.

Example 1. Consider an application that monitors persons and objects as they move through a building using RFID [18]. A user Kate, may want to know the probability that each of her possessions is in her office; she writes a query as in Fig. 1(a). In response to this query, a first-generation system would return a list of her possessions together with a probability that each object is present in her Office (shown in Fig. 1(b)).

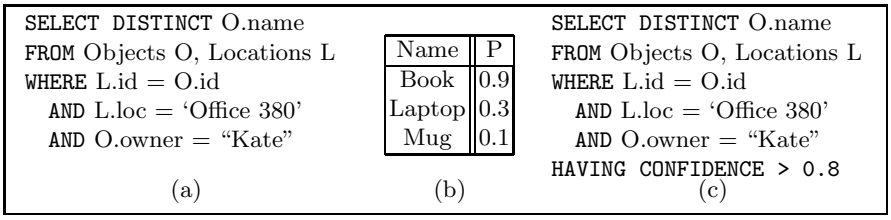

Fig. 1: (a) A first-generation probabilistic query and (b) its output on such a system. Query (c) shows the type of query we study in this paper.

As the research in the area of probabilistic data management has matured, researchers have begun to advocate increasingly sophisticated query languages that allow more complex, direct manipulations of the imprecision in the data [3,9]. A notable feature of these languages is that they allow probability values to be manipulated as first-class citizens. An example of such a query is shown in Fig. 1(c) which asks *"which objects that belong to Kate are in 'Office 380' with probability greater than* 0.8*?"*. These queries are particularly important in applications like RFID event detection, where we want to trigger a real-world action in response to an event (such as sending an email alert when a laptop leaves a room). In addition, the query language that we study captures the essential features of other (more expressive) languages such as recently proposed by Koch [11] or discussed by Fagin et al [7].

In this work, we study the containment problem for conjunctions of queries which are a significant generalization of queries as in Fig. 1(c) (defined formally in § 2.3). Containment is a property of a pair of queries over uncountably many databases, and so it is not immediately clear that the problem is decidable. The main technical contribution of this paper is first showing that containment of such queries is decidable, and further that we can decide this property in EXPSPACE via an algorithm based on Linear Programming.

Outline In §2, we give background on probabilistic databases, provide syntax and semantics for our query language, and formally state our problem. In §3, we show how to construct a linear program that captures the set of databases that will model a particular query. Our main result, the decidability of query containment, is given in §4. We discuss related work (§5) and conclude (§6).

2 Preliminaries

We first give a brief background on probabilistic databases. Then we define our query language and illustrate its use with some examples.

* This work was partially funded by NSF IIS-0713576

2.1 Probabilistic Databases and Orders

A probabilistic database D is a pair $D = (\mathcal{W}, P)$ where $\mathcal{W} = \{W_1, W_2, \ldots, W_n\}$ is a finite set of standard, finite relational databases all conforming to the same schema over an infinite domain $\mathbb{D}$ and $P : \mathcal{W} \rightarrow [0, 1]$ is a distribution function, that is:

$$\sum_{W \in \mathcal{W}} P(W) = 1$$

We refer to $W \in \mathcal{W}$ such that $P(W) > 0$ is a *possible world*.
This definition is very general and contains all (discrete) relational representations in the literature including tuple independent databases [4], x-tables [3], bid tables [15], factor graphs [17], and world-sets [1].
For a given tuple t in the database, the (marginal) probability of a tuple t is denoted $P(t)$ and is defined by

$$P(t) = \sum_{W : W \ni t} P(W)$$

	Name	Office	CurrentLocation
(t_1)	Kate	380	380
(t_2)	Kate	380	Coffee Room
(t_3)	Julie	380	380

Fig. 2: A Standard Relation R. A probabilistic database is a subset of tuples together with a probability function.

Example 2. Fig. 2 shows a standard relational database with a single relation that contains three tuples, $R = \{t_1, t_2, t_3\}$, this database tracks the offices and current locations of various people.
One possible database D is where each tuple is independent with probability p. Our work does not consider tuple independent databases, but we give a brief explanation of them here for comparison. In a tuple independent database, the set of possible worlds is the set of all subsets of tuples in R, that is, $\mathcal{W} = \mathcal{P}(R)$. The probability $P(W)$ of a world W with k tuples (for $k = 1, 2, 3$) is $P(W) = p^k(1 - p)^{3-k}$.

In this work, we view worlds as sets of tuples and then order these worlds by set inclusion (denoted with $\subseteq$). This order is helpful to understand the semantics of the queries that we study in this paper. We do not assume that the tuples in the database are independent, unless explicitly stated. If we arbitrarily assign probabilities to the worlds in this database, we could get many different probability distributions. One possible lattice is depicted in Figure 3.
The result of executing the query, Q(name), on the database shown in Figure 3 is $\{Kate, Julie\}$. However, if the confidence value was 0.8 instead, then the result would be $\{Julie\}$.

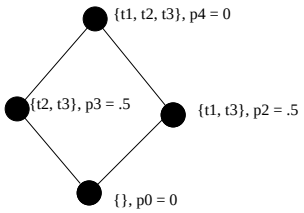

Fig. 3: Lattice Created by Probabilistic Data

```
Q(name) ::=
SELECT DISTINCT R.name
FROM R
WHERE R.currentLocation = '380'
HAVING CONFIDENCE >= .4
```

2.2 Syntax and Semantics of Tier-2 Queries

We consider queries with two tiers: A first tier query is a standard conjunctive Boolean query and is denoted by a lower case q below. The second tier of queries is a set of conjuncts each of which makes a probability statement about first tier queries:

$$q ::= \exists x_1, \exists x_2, \ldots R_1(\bar{x}_1), \ldots R_m(\bar{x}_m)$$
$$Q ::= \exists y_1, \exists y_2, \ldots \exists y_k . P(q_1) \star p_1, P(q_2) \star p_2, \ldots, P(q_n) \star p_n$$

Where $\star$ represents either the $>$ or $\geq$ operator, and $p_i \in (0, 1]$.
Our Tier-2 queries only allow the comparison between the query probability and the values using $>$ or $\geq$. This is essential for the Tier-2 queries to be "monotonic", in a sense that we will make precise below.
As an example, a query that checks for the existence of Kate's book in her office with high probability (similar to the pseudo-SQL query in Fig. 1(c)) is expressed as:

$$\exists i . P\left(\texttt{Object}(i, \text{'Book'}, \text{'Kate'}), \texttt{Location}(i, \text{'Office 380'})\right) > 0.8$$

The semantics of Tier-2 queries requires that we define the semantics of Tier-1 queries. For a Tier-1 query, q, we evaluate q in a given world W as if W is a standard deterministic database. We write $W \models q$ whenever q is true on W.

Definition 1 ($P(q)$). *Given a Tier-1 query q and a probabilistic database $D = (\mathcal{W}, P)$, we define $P(q)$ as*

$$P(q) = \sum_{W \in \mathcal{W} : W \models q} P(W)$$

Definition 2 (Tier-2 Semantics). *Given a Tier-2 query Q and a probabilistic database $D = (\mathcal{W}, P)$, we say that Q is true on D, and denote $D \models Q$, if there exists a valuation $\theta : \{y_1, \ldots, y_k\} \to \mathbb{D}$ such that for each $i = 1, \ldots, n$ then $P(q_i[\theta(\overline{y})/\overline{y}]) \geq p_i$ where $q_i[\theta(\overline{y})/\overline{y}]$ denotes the query that results when each y_i is substituted with $\theta(y_i)$.*

Thus, Tier-2 queries are for probabilistic databases what standard conjunctive queries (called here Tier-1 queries) are for standard databases. Tier-2 queries are also "monotone". To make this statement precise, we first need to define an order relation on probabilistic databases.

Definition 3 ($F(W, D)$). *Given a probabilistic database $D = (\mathcal{W}, P)$, and a world $W \in \mathcal{W}$, we define the filter of W, written $F(W, D)$ as:*

$$F(W, D) = \{W' \in D | W \subseteq W'\}$$

Definition 4 ($D \leq D'$). *Given two probabilistic databases D and D' over the same set of possible worlds $\mathcal{W}$, we define the order $D \leq D'$ if for all $W \in \mathcal{W}$, $P(F(W, D)) \leq P(F(W, D'))$.*

The next Proposition shows that Tier-2 queries are monotone. The proofs of the Proposition and the Lemma are straightforward and omitted.

Lemma 1. *If $D \leq D'$ and q is a Tier-1 query, then $P(q) \leq P'(q)$.*

Proposition 1. *If $D \leq D'$ and Q is a Tier-2 query, then $D \models Q \implies D \models Q'$*

2.3 Problem Statement and Main Result

We now have all the necessary notation to define our problem formally:

Definition 5 (Tier-2 Query Containment Problem). *Let Q and Q' be Tier-2 Queries, we say that Q is contained in Q' if for* any *probabilistic database D, the following holds:*

$$D \models Q \implies D \models Q'$$

The Tier-2 Query containment Problem is to decide, given as input two Tier-2 queries Q and Q', is Q contained in Q'?

Since containment is a property of an infinite set of databases, it is not immediately clear that containment is decidable. The main result of this work is that it is decidable (in EXPSPACE) using the algorithm of §4.

2.4 Examples

To assist with the reader's understanding of this problem, we introduce two small examples and answer the question: Is Q contained in Q'?

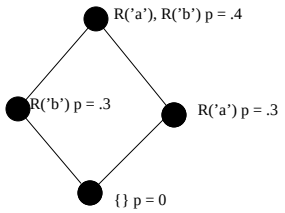

Fig. 4: $D \models Q$

Example 3.

$$Q = P(R('a')) \geq .7, P(R('b')) \geq .7$$
$$Q' = P(R('a'), R('b')) \geq .4$$

To answer the containment question, we first construct a database, D, that satisfies Q. A picture of such a database is shown in Figure 4.
Now, it is also clear that $D \models Q'$, but are there any other databases, D' such that $D' \models Q$ but $D' \not\models Q'$? No. To explain this, we present the following argument. Q has two parts; one of which requires a world where $R('a')$ is true, and one where $R('b')$ is true. If we allow p_1 to represent the probability of all worlds that contain 'a' but not 'b', p_2 to represent the probability of all worlds that contain 'b' but not 'a', and p_3 to represent the probability of all worlds that contain both 'a' and 'b', then the following restrictions hold:

$$p_1 + p_3 \geq 0.7$$
$$p_2 + p_3 \geq 0.7$$

If $p_3 < 0.4$ then it must also be the case that $p_1 > .3$. If this were true, we would have that $p_1 + p_2 + p_3 > 0.3 + 0.7 = 1$. However, this is an invalid database, and thus we reach a contradiction.

Example 4.

$$Q = P(\exists x.R(x)) \geq 1$$
$$Q' = \exists y.P(R(y)) \geq .5$$

Figure 5 shows two different databases, where the one on the left is such that $D \models Q$ and $D \models Q'$. However, the database on the right serves as a counterexample to the containment because for this one, $D' \models Q$, but $D' \not\models Q'$.

3 Tier-2 Queries, Q, as Linear Programs

In this section, we start from a 2nd tier query Q, and then we define a *set of canonical linear programs* for that query, called $SLP(Q)$. We

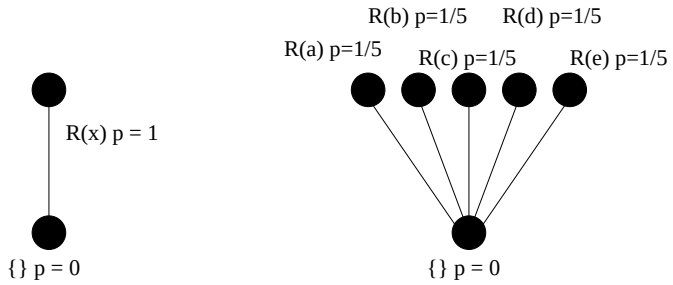

Fig. 5: Two related databases with different results.

prove that $SLP(Q)$ captures the semantics of Q. Because the semantics of Q is defined over probabilistic databases with arbitrarily large sets of worlds, the number of probability values $p_1, p_2, p_3, \ldots$ in the probabilistic database can be arbitrarily large: on the other hand, $SLP(Q)$ has a *fixed* set of variables that only depends on the query, and not on a database. This allows us to use $SLP(Q)$ to concisely capture the semantics of Q, because it makes a statement about arbitrarily many probabilistic values.

3.1 Queries without 2nd-Tier Quantifiers

When the query has no second tier quantifiers, then $SLP(Q)$ is a single linear program, which we denote $LP(Q)$. Let

$$Q = P(q_1) \geq p_1, \ldots P(q_n) \geq p_n$$

$LP(Q)$ has variables $\boldsymbol{v}$ that correspond to equivalence classes of conjuncts of q_i. Fig. 6 shows the algorithm that constructs $LP(Q)$. In this program, a variable v_X represents the probability mass assigned to the worlds W such that the sum of the mass of all worlds W where $W \models q_i, i \in X \geq p_i$.

Definition 6. *Let Q be a tier-2 query and $LP(Q)$ its associated linear program (Fig. 6), then for any world W let $\Sigma(W) = \{i | W \models q_i\}$ (note that $\Sigma(W)$ is in $\mathfrak{C}$). For any probabilistic database $D = (\mathcal{W}, P)$ the solution associated with D is denoted $\boldsymbol{v}(D)$ and is an assignment to the variables of $LP(Q)$ defied by*

$$v_X = \sum_{W:\Sigma(W)=X} P(W)$$

Theorem 1. *Let $Q = P(q_1) \geq p_1, \ldots, P(q_n) \geq p_n$, $LP(Q)$ be as defined in Fig. 6, and $D = (\mathcal{W}, P)$. Then, the following two statements are equivalent:*
(1) $D \models Q$
(2) $\boldsymbol{v}(D)$, as defined in Definition 6, is a feasible solution to $LP(Q)$.

The theorem follows from the following lemma:

Input: A query $Q = P(q_1) \geq p_1, \ldots P(q_n) \geq p_n$
Returns: A Linear Program $LP(Q)$

---

1. For every $X \subseteq \{1, \ldots, n\}$, define a conjunctive query:

$$q_X \stackrel{def}{=} \bigwedge_{i \in X} q_i$$

2. For $X \subseteq \{1, \ldots, n\}$, define its closure $X^* = \{i | q_X \subseteq q_i\}$. Note that $q_X \equiv q_{X^*}$. Let $\mathfrak{C}$ be the set of closed sets.
3. For each $X \in \mathfrak{C}$, we create a variable v_X.
4. We then add the following linear constraints over v_X.
   - For every $X \in \mathfrak{C}$, add a constraint $v_X \geq 0$.
   - Add the constraint $\sum_{X \in \mathfrak{C}} v_X = 1$.
   - For every q_i in Q, add $\sum_{X \in \mathfrak{C}: X \ni i} v_X \geq p_i$

Fig. 6: The algorithm to construct $LP(Q)$

Lemma 2. *Let $D = (\mathcal{W}, P)$ be a probabilistic database and let $\boldsymbol{v}(D)$ be its associated solution then, for all i*

$$\sum_{X: X \ni i} v_X = \sum_{W: W \models q_i} P(W)$$

Proof. It suffices to observe that $W \models q_i$ if and only if $\Sigma(W) \ni i$.

The theorem follows since each database D such that $D \models Q$ is a feasible solution. On the other hand, given a feasible solution $\boldsymbol{v}$, we take $\mathcal{W} = \{I_X \mid q_X\}$ where I_X is any canonical instance for the conjunctive query q_X and $P(I_X) = v_X$.
An example of how to construct $LP(Q)$ is given in the Appendix.

3.2 Queries with 2nd-Tier Quantifiers

In the prior section, we handled only the case where Q did not have any existential quantifiers at the Tier-2 level. We address this limitation here and show how to construct $SLP(Q)$ whenever Q is of the form:

$$Q = \exists y_1, \exists y_2, \ldots, \exists y_k . P(q_1) \geq p_1, \ldots P(q_n) \geq p_n$$

We define some sets:

$$C = \{c | c \text{ is a constant that appears in one of the } q_i\}$$
$$C' = \{c_1 \ldots c_k | c_i \text{ is a fresh constant }\}$$
$$CONST = C \cup C'$$

Then for each mapping, $\theta : \{y_1, \ldots, y_k\} \rightarrow CONST$, θ provides a substitution for the y's in Q to produce $\hat{Q}_\theta$:

$$\hat{Q}_\theta = P(q_1[\theta(\overline{y})/\overline{y}]) \geq p_1, \ldots P(q_n[\theta(\overline{y})/\overline{y}]) \geq p_n$$

Define $\mathcal{L}_\theta$ as the the linear program that corresponds to $\hat{Q}_\theta$.

Theorem 2. *Let $D = (\mathcal{W}, P)$, then the following statements are equivalent:*
- $D \models Q$
- $\exists \theta$ *such that* $\boldsymbol{v}(D)$ *is a feasible solution for* $\mathcal{L}_\theta$

Proof (Theorem 2). The proof of this theorem follows directly from the proof of Theorem 1, and from the semantics of second tier queries given in Section 2.2.

All possible mappings and their corresponding LPs are added to $SLP(Q)$. An example of how to construct $SLP(Q)$ is given in the Appendix.

4 Query Containment

In this section, we show that Query Containment for conjunctive, tier-2 queries is decidable. There are two tier-2 queries as input:

$$Q = \exists y_1, \ldots, y_k . P(q_1) \geq p_1, \ldots P(q_n) \geq p_n$$
$$Q' = \exists y_1, \ldots, y_l . P(q'_1) \geq p'_1, \ldots P(q'_m) \geq p'_m$$

4.1 Preliminaries

The key technical hurdle is illustrated by trying to decide containment for the following pair of queries.

Example 5. Let $Q = P(\exists x . R(x)) \geq 1$ and $Q' = \exists y . P(R(y)) \geq N^{-1}$ for some N. Intuitively, Q states that the relation R is not empty, while the query Q' says that some particular value is present with probability at least N^{-1}. Example 4 is very similar to this one, but less general. There is no containment relationship between these two queries, but providing a counterexample is surprisingly subtle. In particular, we want to a provide a counterexample database D such that $D \models Q$, but $D \not\models Q'$. One such counterexample D is a database with $N+1$ worlds $I_i = \{R(c_i)\}$ for $i = 1, \ldots, N+1$ and $P(I_i) = (N+1)^{-1}$. This is somewhat jarring because we seem to need to make $N+1$ copies of the same database. A little thought shows that this construction is necessary, *any counter example must have at least* $N+1$ *worlds*. This is so that we can drive down the importance of any particular evaluation of the constants.

The proof of the main containment theorem expands on this idea. To do so, we need to be precise by what we mean about copying. Thus, we introduce a little bit of notation. A *canonical world* for a conjunctive query $q = g_1, \ldots, g_n$ is a world I such that there exists a bijective homomorphism $h : \mathbf{var}(q) \to \mathbb{D}$ such that $I = \bigcup_{i=1,\ldots,m} h(g_i)$. For example, $R(a,b)$ and $R(b,c)$ are both canonical worlds for $q = \exists x . R(x,y)$, but $R(a,a)$ is not.
Given a query $q'(\boldsymbol{y})$ where y denotes the head variables of q' (thus q' is not necessarily a boolean query), and a query q, let $\text{Cert}(q', q) = \{\boldsymbol{t} \mid q \subseteq q'(\boldsymbol{t})\}$, i.e., those values of t that are certain answers of q' on the canonical world for q. Denote $\text{Cert}(Q', q) = \bigcup_{q' \in Q'} \text{Cert}(q', q)$.

Lemma 3. *Given a Boolean conjunctive query, q_i, with at least a single variable, there is an infinite set of canonical worlds for q_i, $\{I_1, \ldots, I_n, \ldots\}$ such that for* any *conjunctive query $q'_j(\boldsymbol{y})$ and any $\boldsymbol{t}$ of the same arity as the head of q'_j exactly one of the following two conditions holds:*
(1) $\boldsymbol{t} \in \text{Cert}(q'_j, q_k)$ and so $I_i \models q'_j(\boldsymbol{t})$ for each i, or otherwise
(2) there is at most one I_i such that $I_i \models q'_j(\boldsymbol{t})$.

Proof. To construct the desired set of worlds, simply map each variable in q_i to a distinct, fresh variable in each world. This set of worlds has the desired property.

We call the set I_i for $i = 1, 2, \ldots$ the set of copies for q_i. Notice that its construction does *not* depend on q'_j.

4.2 Main Result

Consider the case when there are no Tier-2 quantifiers in Q (y variables). We will remove this restriction later in the section.

The Linear Program We construct a set of canonical linear programs $CLP(Q)$: there is one program for each $q'_j \in Q'$. Consider some fixed $q'_j \in Q'$ with probability p'_j. The program for q'_j has the same variables as $LP(Q)$, and all constraints of $LP(Q)$. In addition, we add a constraint for each $t_0 \in \text{Cert}(Q', q)$ of the following form:

$$\sum_{X \in \mathfrak{C}: q_X \subseteq q'_j(t_0)} v_X < p'_j$$

An example of how to construct $CLP(Q)$ is given in the Appendix.
This program captures containment in the following sense:

Theorem 3. *There is a feasible solution to some program in $CLP(Q, Q')$ if and only if Q is not contained in Q'.*

Proof. Suppose there is a feasible solution, we show how to construct a probabilistic database $D = (\mathcal{W}, P)$ that is a counterexample to containment. Note that what is needed here is that such a counterexample exists; it is constructed only for the sake of the proof.
First, given a feasible solution $\boldsymbol{v}$, we define $\varepsilon(\boldsymbol{v})$ to be the smallest slack in the linear program. More precisely,

$$\varepsilon(\boldsymbol{v}) = \min_{t_0 \in \text{Cert}(Q', q)} p'_j - \left(\sum_{X \in \mathfrak{C}: q_X \subseteq q'_j(t_0)} v_X \right)$$

There are finitely many constraints, and the slack is non-zero in each constraint, so we have that $\varepsilon(\boldsymbol{v}) > 0$. Choose N such that $N^{-1}2^n = \varepsilon/2$ where n is the number of q_i queries in Q.
For each q_X, we generate N canonical instances for q_X that we denote with $\mathcal{W}_X = \left\{I_{(X,1)}, \ldots, I_{(X,N)}\right\}$. Here, we choose any N worlds from the

copy worlds defined above. We let $\mathcal{W} = \bigcup_X \mathcal{W}_X$. We set the probability function P as $P(I_{X,i}) = v_X N^{-1}$, that is the mass for v_X is evenly distributed among all its copies.
This must be a counterexample because on the database D, $\exists y.P(q'_j(y)) < p'_j$ holds. Indeed, for a fixed t_0 we can write:

$$\begin{aligned} P(q'_j(t_0)) &\leq \sum_{X \in \mathfrak{C}: q_X \subseteq q'_j(t_0)} v_X + \sum_{Y \in \mathfrak{C}: q_Y \not\subseteq q'_j(t_0)} N^{-1} \\ &< p' - \varepsilon + 2^N 2^{-N} \varepsilon/2 < p' \end{aligned}$$

The first line follows by construction: each copy I corresponding to an X in the first sum is such that $I \models q'_j(t_0)$. Thus, for any choice of t_0 it must be that $q'_j(t_0)$ is satisfied in at most one world, and hence with probability $\leq N^{-1}$. This follows as a result of Lemma 3 and the fact that there are at most 2^N such sets, Y. Then, by our selection of N and the definition of $\varepsilon(\boldsymbol{n})$, we have the second line. Since our choice of t_0 was arbitrary, this holds for all t_0 simultaneously. Thus, we have that $q'_j(t_0) < p'_j$ for any t_0. Finally, we note that $\boldsymbol{v}$ is a feasible solution to $LP(Q)$ as well, and hence by Thm. 1, $D \models Q$. Thus, we have produced a counterexample to containment.
Now, we prove the other direction. Assume that we have any counterexample database D. Construct $\boldsymbol{v}(D)$, which is an assignment of probabilities to worlds in D. Observe:

$$P(q'_j(t_0)) \geq \sum_{X \in \mathfrak{C}: q_X \subseteq q'_j(t_0)} v_X$$

Thus, for each t_0 we have $P(q'_j(t_0)) < p'$. Since $D \models Q$, this solution also satisfies $LP(Q)$. Hence, a counterexample implies a feasible solution.

Q Contains Quantifiers We guess a mapping θ from (the Tier-2 quantifiers in Q) to constants so that $\theta : \{y_1, \ldots, y_k\} \rightarrow CONST$ and then apply the previous theorem. This gives a simple NEXPSPACE algorithm and then by Savitch's theorem:

Corollary 1. *We can decide containment in* EXPSPACE

5 Related Work

As we mentioned in the introduction, there is a large body of work in probabilistic databases [15,19] that deals with how to represent and store probabilistic data. The first generation of probabilistic databases mainly varied in their ability to represent distributions succinctly including tuple independent databases [4], x-tables [16], and more succinct representations such as factor graphs [17].
Recently, there has been a renewed interest in query languages for probabilistic data from the MayBMS group [1,9]; their query language is fully compositional. As a result, their language can be viewed as allowing an

arbitrary number of tiers as opposed to only two. They do not study the containment problem.
There has also been renewed interest in containment. Notably the breakthrough results of Green [10], who showed containment results for queries over relations annotated with elements from a semiring. The language study in his work is a classical language similar to a first-generation probabilistic database: it does not manipulate the annotations within the query language. It is an interesting extension of our work to study containment for a rich query language that manipulates (more general) semiring annotations.
Fagin et. al. [7] study a language for reasoning about probability that is similar to the one we present here. They also demonstrate that systems of linear inequalities can be used to capture the set of probability spaces that will satisfy a particular statement in their language. Additionally, their result states that when the queries are restricted to propositional logic the construction is in NP. However, this work does not consider query containment.

6 Conclusions and Future Work

We presented a decidable characterization of containment for a rich query language over probabilistic databases, which is the first such algorithm to address query containment of a probabilistic language that allows direct manipulation of uncertainty. The future work for this project is in three directions: (1) more expressive annotations and query languages, such as arbitrary levels of nesting, (2) we plan to study query containment over restricted, but practically important, classes of probabilistic databases, such as tuple independent probabilistic databases, and (3) we plan to find tractable subclasses of queries for the containment problem.

References

1. Lyublena Antova, Christoph Koch, and Dan Olteanu. 10^{10^6} worlds and beyond: Efficient representation and processing of incomplete information. In *ICDE*, pages 606–615, 2007.
2. Lyublena Antova, Christoph Koch, and Dan Olteanu. Maybms: Managing incomplete information with probabilistic world-set decompositions. In *ICDE*, pages 1479–1480, 2007.
3. Omar Benjelloun, Anish Das Sarma, Alon Halevy, Martin Theobald, and Jennifer Widom. Databases with uncertainty and lineage. *The VLDB Journal*, 17(2):243–264, 2008.
4. Nilesh N. Dalvi and Dan Suciu. Efficient query evaluation on probabilistic databases. In *VLDB*, pages 864–875, 2004.
5. Yanlei Diao, Boduo Li, Anna Liu, Liping Peng, Charles Sutton, Thanh Tran 0002, and Michael Zink. Capturing data uncertainty in high-volume stream processing. In *CIDR*, 2009.
6. Xin Luna Dong, Alon Y. Halevy, and Cong Yu. Data integration with uncertainty. In *VLDB*, pages 687–698, 2007.

7. Ronald Fagin, Joseph Y. Halpern, and Nimrod Megiddo. A logic for reasoning about probabilities. *Inf. Comput.*, 87(1/2):78–128, 1990.
8. Avigdor Gal, Maria Vanina Martinez, Gerardo I. Simari, and V. S. Subrahmanian. Aggregate query answering under uncertain schema mappings. In *ICDE*, pages 940–951, 2009.
9. Michaela Götz and Christoph Koch. A compositional framework for complex queries over uncertain data. In *ICDT*, pages 149–161, 2009.
10. Todd J. Green. Containment of conjunctive queries on annotated relations. In *ICDT*, pages 296–309, 2009.
11. Christoph Koch. On query algebras for probabilistic databases. *SIGMOD Record*, 37(4):78–85, 2008.
12. Imran R. Mansuri and Sunita Sarawagi. Integrating unstructured data into relational databases. In *ICDE*, page 29, 2006.
13. Christopher Re, Nilesh N. Dalvi, and Dan Suciu. Efficient top-k query evaluation on probabilistic data. In *ICDE*, pages 886–895, 2007.
14. Christopher Ré, Julie Letchner, Magdalena Balazinska, and Dan Suciu. Event queries on correlated probabilistic streams. In *SIGMOD Conference*, pages 715–728, 2008.
15. Christopher Re and Dan Suciu. Materialized views in probabilistic databases for information exchange and query optimization. In *VLDB*, pages 51–62, 2007.
16. Anish Das Sarma, Martin Theobald, and Jennifer Widom. Exploiting lineage for confidence computation in uncertain and probabilistic databases. In *ICDE*, pages 1023–1032, 2008.
17. Prithviraj Sen and Amol Deshpande. Representing and querying correlated tuples in probabilistic databases. In *ICDE*, pages 596–605, 2007.
18. University of Washington. RFID Ecosystem. http://rfid.cs.washington.edu/.
19. Jennifer Widom. Trio: A system for integrated management of data, accuracy, and lineage. In *CIDR*, pages 262–276, 2005.
20. Yahoo! Research. The Purple Sox System. http://research.yahoo.com/node/498.

A Appendix: Examples

To provide the reader with more intuition regarding how to construct $LP(Q)$, $SLP(Q)$, and $CLP(Q)$, we present a complete example here.

A.1 Constructing $LP(Q)$

Let

$$Q = P(R('a')) \geq .2, P(R('b')) \geq .8$$

Because Q has only constants we can construct a single linear program, $LP(Q)$. First, we observe that

$$q_1 = R('a') \text{ and } q_2 = R('b')$$

In this case there are no shared tuples between q_1 and q_2, so the linear program we construct will have four variables, $v_{\{\}}, v_{\{1\}}, v_{\{2\}}$, and $v_{\{1,2\}}$. Where $v_{\{\}}$ represents the probability mass assigned to the canonical world for an "empty query", $v_{\{1\}}$ represents the probability mass assigned to the canonical world for q_1, $v_{\{2\}}$ represents the probability mass assigned to the canonical world for q_2, and $v_{\{1,2\}}$ represents the probability mass assigned to the canonical world for $q_1 \wedge q_2$.
From this, we get the following LP:

Non-Negativity Constraints	$v_{\{\}}, v_{\{1\}}, v_{\{2\}}, v_{\{1,2\}} \geq 0$
Validity Constraint	$v_{\{\}} + v_{\{1\}} + v_{\{2\}} + v_{\{1,2\}} = 1$
Constraint for q_1	$v_{\{1\}} + v_{\{1,2\}} \geq .2$
Constraint for q_2	$v_{\{2\}} + v_{\{1,2\}} \geq .8$

A.2 Constructing $SLP(Q)$

Now, let

$$Q = \exists y_1, y_2.P(R(y_1)) \geq .2, P(R(y_2)) \geq .8$$

In this case, there are two mappings we need to consider. The first mapping (θ_1) gives different constants for y_1 and y_2, whereas the second mapping (θ_2) yields the same constant for y_1 and y_2.

θ_1 mapping Here, we assume that $\theta_1(y_1) \neq \theta_1(y_2)$. Since the actual values of the constants can be chosen arbitrarily, assume that

$$\theta_1(y_1) =' a'$$

$$\theta_1(y_2) =' b'$$

In this case,

$$\hat{Q}_{\theta_1} = P(R('a')) \geq .2, P(R('b')) \geq .8$$

As with the prior example, we have four "canonical worlds", and thus $\mathcal{L}_{\theta_1}$ is:

Non-Negativity Constraints	$v_{\{\}}, v_{\{1\}}, v_{\{2\}}, v_{\{1,2\}} \geq 0$
Validity Constraint	$v_{\{\}} + v_{\{1\}} + v_{\{2\}} + v_{\{1,2\}} = 1$
Constraint for q_1	$v_{\{1\}} + v_{\{1,2\}} \geq .2$
Constraint for q_2	$v_{\{2\}} + v_{\{1,2\}} \geq .8$

θ_2 mapping This time, we assume that $\theta_2(y_1) = \theta_2(y_2)$. Since the actual values of the constants can be chosen arbitrarily, we simply choose

$$\theta_2(y_1) = \theta_2(y_2) =' a'$$

This mapping provides a different query:

$$\hat{Q}_{\theta_2} = P(R('a')) \geq .2, P(R('a')) \geq .8$$

Here, $n = 2$ and of the four subsets of $\{1, 2\}$ only two are closed: $\{\}$ and $\{1, 2\}$. Thus, this query yields only 2 canonical worlds (the one that contains $R('a')$ and the empty world), and only two variables are

needed for $\mathcal{L}_{\theta_2}$. We construct a new linear program:

Non-Negativity Constraint	$v_{\{\}}, v_{\{1,2\}} \geq 0$
Validity Constraint	$v_{\{\}} + v_{\{1,2\}} = 1$
Constraint for q_1	$v_{\{1,2\}} \geq .2$
Constraint for q_2	$v_{\{1,2\}} \geq .8$

$SLP(Q) = \{\mathcal{L}_{\theta_1}, \mathcal{L}_{\theta_2}\}$.

A.3 Constructing $CLP(Q)$

For this step, we need two queries, Q and Q'. Let

$$Q = P(R('a')) \geq .2, P(R('b')) \geq .8$$

and

$$Q' = \exists y_1, y_2 . P(R(y_1), R(y_2)) \geq .3, P(R('b')) \geq .4$$

We also assume that we are given $LP(Q)$ (which was constructed in an earlier section of this example). In order to check containment, we want to know if there are any databases that will satisfy Q but not Q'. Since Q' is a conjunct of two sub-queries, we only need a solution that finds a counterexample for one of our two sub-queries.

Sub-query $q_1 = P(R(y_1), R(y_2)) \geq .3$ If we evaluate the arity-2 query $R(y_1), R(y_2)$ on the canonical worlds from the $LP(Q)$, we get only one world, the one that contains both $R('a')$ and $R('b')$. The linear program for Q uses variable $v_{\{1,2\}}$ to represent the probability mass assigned to this world, so we add a new constraint to $LP(Q)$ to get an LP for containment. The new program is:

$LP(Q)$	$\vdots$
$\neg q_1$ Constraint	$v_{\{1,2\}} < .3$

This program is added to $CLP(Q)$.

Sub-query $q_2 = P(R('b')) \geq .4$ If we evaluate the arity-1 query $R(y_1)$ on the canonical worlds from the $LP(Q)$, we get two worlds, the one that satisfies $R('b')$ only and the one that contains both $R('a')$ and $R('b')$. The linear program for Q uses variables $v_{\{2\}}$ and $v_{\{1,2\}}$ to represent the probability mass assigned to these worlds (respectively). We add a new constraint to $LP(Q)$ to get an LP for containment. This new program is:

$LP(Q)$	$\vdots$
$\neg q_2$ Constraint	$v_{\{2\}} + v_{\{1,2\}} < .4$

This program is also added to $CLP(Q)$, which (now complete) contains two linear programs, one for each sub-query of Q'.
The first LP has a feasible solution, and thus provides a counter-example to containment, whereas the second LP has no feasible solution.