Query Evaluation with Soft-Key Constraints

Abhay Jha
CSE, University of Washington
Seattle WA, 98195-2350
abhaykj@cs.washington.edu

Vibhor Rastogi
CSE, University of Washington
Seattle WA, 98195-2350
vibhor@cs.washington.edu

Dan Suciu
CSE, University of Washington
Seattle WA, 98195-2350
suciu@cs.washington.edu

ABSTRACT

Key Violations often occur in real-life datasets, especially in those integrated from different sources. Enforcing constraints strictly on these datasets is not feasible. In this paper we formalize the notion of soft-key constraints on probabilistic databases, which allow for violation of key constraint by penalizing every violating world by a quantity proportional to the violation. To represent our probabilistic database with constraints, we define a class of markov networks, where we can do query evaluation in PTIME. We also study the evaluation of conjunctive queries on relations with soft keys and present a dichotomy that separates this set into those in PTIME and the rest which are #P-Hard.

1. INTRODUCTION

Soft constraints are emerging as a promising approach to cope with various kinds of uncertainty in data, as found in many modern applications. Soft constraints have been used to enhance the quality of information extraction [25], of object reconciliation [26, 18], in query optimization [16], and in data cleaning [1].

While discovering soft constraints is possible today given advances in machine learning, using the soft constraints during query processing over large volumes of data is much harder. Current approaches to probabilistic inference are based either on Monte Carlo Markov Chain [22] or on message passing [7], and these do not scale to large volumes of data. The main problem that prevents us from adopting soft constraints is the lack of scalable query processing techniques in the presence of soft constraints.

In this paper we study *soft key constraints*, or *soft keys* in short, and examine the query evaluation problem: evaluate a Boolean conjunctive query on a database given a set of soft keys. We interpret soft keys using Markov Networks whose potential consists of two parts, one that depends on individual tuples, and the other that depends only on the number of tuples that have the same key; such potentials have been recently studied in [15]. Our soft keys are in fact general cardinality constraints. We define query evaluation as computing the marginal probability of the query, which is the common semantics in probabilistic databases [3, 12, 5, 9]: this is different from computing the most likely world, a problem studied in [15].

While general query evaluation on probabilistic database is known to be #P-hard, even without soft keys [9], we identify two cases that are tractable and describe two polynomial time algorithms for evaluating conjunctive queries in the presence of soft key constraints: to our knowledge these are the first provably tractable algorithms in the presence of any soft constraints. Our first algorithm applies to queries over a single relation, in the presence of multiple soft keys. Our second algorithm applies to conjunctive queries over multiple relations, with multiple soft keys. In both cases we also establish a dichotomy: if our algorithms do not apply then we can show, under certain assumptions, that the query is #P-hard.

Our analysis is similar in spirit to a previously known dichotomy result for query evaluation on disjoint/independent probabilistic databases [9], which can be thought of as probabilistic databases with hard key constraints. That result defines a syntactic condition on the query, called *safety*, then proves that every safe query can be evaluated in PTIME and that every unsafe query is #P-hard. In our work we also define a syntactic *safety* condition for queries in the presence of soft keys. Then we show that a query is safe in the presence of soft keys iff it remains safe after making every key either hard, or removing it altogether, in all possible ways. The intuitive significance of this connection is the following: by varying the weight attached to a soft key one can either make it hard or remove it completely, and therefore any PTIME algorithm that can handle soft keys needs to be able to handle these extremes. However, the PTIME algorithm that we describe in this paper in the presence of soft keys is significantly more difficult than the previous algorithm for safe queries. This is by necessity: even one soft key makes query evaluation much more difficult, and the interaction between multiple soft keys on the same table adds even more complexity.

1.1 Motivating Example

To motivate our work, we illustrate with a concrete problem: duplicate elimination in dirty data. This occurs frequently in data integration because of different representation conventions, or simply because of typos, and results in key violations: a person has multiple addresses, a company has multiple CEO's, a scientific paper has multiple years of publication. Andritsos et al. [1] have proposed a probabilistic approach to answer queries directly on the dirty data. They assign to each duplicate tuple a probability, such that the probabilities for the same key sum up to 1. For example, consider a relation `Person(name, city)`, where we define `name` to be a key. If we find two tuples with the same `name`, say `(Joe, Seattle)` and `(Joe, Whistler)` then we have a key violation: the approach in [1] is to assign to each tuple a probability, say 0.5, indicating that only one tuple may be present in a clean instance. While this approach uses probabilities, the key constraint is *hard*: in each possible world only one of the two tuples

`Person:`

	`Name`	`City`	`W`
t_1	Joe	Seattle	$w_1 = 4$
t_2	Joe	Whistler	$w_2 = 3$
t_3	Frank	Seattle	$w_3 = 0$
t_4	Frank	Paris	$w_4 = -2$
t_5	Frank	Honolulu	$w_5 = 1$
t_6	Sue	Portland	$w_6 = 0$
t_7	Sue	Whistler	$w_7 = -1$
t_8	Lisa	Paris	$w_8 = 3$
t_9	Lisa	Milan	$w_9 = -1$
t_{10}	Lisa	Saint Malo	$w_{10} = -1$

Figure 1: A probabilistic relation given by weights

```
SOFT KEY x ON Person(x,y)
    SIZE 2 WEIGHT -4;
SOFT KEY ON Person(Frank,y)
    SIZE 2 WEIGHT 4
    SIZE 3 WEIGHT -3;
SOFT KEY x WHERE Income(x,z), z > 1M
    ON Person(x,y)
    SIZE 2 WEIGHT 3
    SIZE s WHERE s > 2 WEIGHT -2*s
```

Figure 2: Soft key constraints for `Person`

may be present.

With a *soft key*, we can relax the constraint, by allowing a possible world to contain multiple occurrences of the same key, but by assigning a certain penalty for multiple occurrences. There are two reasons why we need such a relaxation. First is that constraints are learned from training data, rather than stated by an administrator, and these are always "soft". For example, in the case of historical data, a person's address is not unique at all ! A machine learning tool may infer, for example, that people over 50 years old typically have 6 addresses, because they moved in the past, while people under 25 typically have 1 address. Such detailed statistical information about the data can be represented using our notion of soft keys.

Second, by ignoring the "softness" of a key, we get wrong query answers. For example, assume that most people have a single residence, but wealthy people may have a second vacation home, and perhaps a third small apartment in a big city. Consider a user who integrates `Person(name, city)` with `Hobby(name, hobby)`, searching for cities likely to host skiers:

```
q(y) :- Person(x,y), Hobby(x, 'Ski')
```

`Whistler` is a popular ski resort in Canada, but very few people actually live there. However, many wealthy people have a condo or a vacation home in `Whistler`. With a hard key constraint, all these entries appear to the system to be wrong, and it will decrease their probabilities: other cities will rank higher than `Whistler`, only because the system found fewer key violations for those cities. In fact, a probabilistic database system that returns only the top k most likely answers [19] may not retrieve `Whistler` at all.

Our approach is to use soft keys instead of hard keys; we illustrate it in Figure 1. `Person` is a probabilistic database [9], where instead of probabilities we indicate the weight of each tuple[1]. As in [1], we can assign different weights to different tuples to indicate our belief in the quality of the data that produced each tuple. However, unlike [1] tuples with the same key are not disjoint: keys are "soft". In our approach the soft keys are specified separately, using a declarative language that is illustrated in Fig 2. The first constraint says that two cities for the same person should be penalized with a weight $w = -4$. The second constraint defines two soft keys. One says that `Frank` is allowed to have two cities: its weight $w = 4$ cancels the weight of the first soft key. The second penalizes with a weight -3 if 3 tuples are found with the key `Frank`. The next soft key says that wealthy people are actually more likely to have two homes, then gives a formula on how to decrease their weight as function of the number of homes. Together, these soft keys capture our statistical knowledge about the data, and need to be used during query evaluation. For example, the query q above will return `Whistler` with a much higher probability, because the wealthy people that have vacation homes are no longer considered errors by the system.

[1]The weight w and the probability p are generally related through $w = \log \frac{p}{1-p}$, but see Examples 2.6 and 2.7.

Organization We give the basic definitions and the background on Markov Networks in Sec. 2, then study the query evaluation problem on a single relation in Sec. 3. We show how to extend it to conjunctive queries over multiple relations and establish the dichotomy in Sec. 4.

2. DEFINITIONS AND NOTATIONS

We begin with a brief review of Markov networks (for a detailed description, we refer to [17, 7]). Then, we define a particular kind of a Markov Network to interpret the soft keys.

2.1 Markov Networks

A Markov Network is a concise presentation of a probability distribution of a set of random variables $\bar{X} = \{t_1, t_2, \ldots, t_n\}$. A *state* is defined to be an assignment to the variables in $\bar{X}$. In this paper we restrict to Boolean variables, and therefore we will assimilate a state with a subset $W \subseteq \bar{X}$, and call it a *world*. The set of possible worlds is $\mathcal{W} = 2^{\bar{X}}$. Thus, a probability distribution on the random variables $\bar{X}$ is a finite probability space $(\mathcal{W}, P)$, where $\mathcal{W} = 2^{\bar{X}}$ and $P : \mathcal{W} \to [0, 1]$ s.t. $\sum_{W \in \mathcal{W}} P(W) = 1$; the atomic, exclusive events of the probability space are the *possible worlds* $W \in \mathcal{W}$.

Typically, n is very large: in a probabilistic database each tuple corresponds to a Boolean variable t_i (hence we will refer to t_i as a tuple) and a world corresponds to a subset of tuples. Thus, n is the number of tuples in the probabilistic database. It is not possible to enumerate all 2^n values of P.

A Markov Network (MN) describes the function P more concisely. The MN is a triple $(\bar{X}, K, (\Phi_c)_{c \in K})$, where K is a set of subsets of $\bar{X}$, called *cliques*, and for each $c \in K$, $\Phi_c : \{0, 1\}^c \to \mathbb{R}^+$ is called a *potential function*. The probability distribution $P : \mathcal{W} \to [0, 1]$ defined by the MN is $P(W) = \frac{1}{Z}\Phi(W)$, where $\Phi(W)$ is called the weight of the world W and Z is a normalization factor, and are given by:

$$\Phi(W) = \prod_{c \in K} \Phi_c(W \cap c)$$
$$Z = \sum_{W \in \mathcal{W}} \Phi(W)$$

Thus, instead of having to enumerate 2^n values for P, we can just enumerate $2^{|c|}$ values of each potential Φ_c. The basic assumption in Markov Networks is that each clique c is small.

An MN defines an undirected graph with nodes $\bar{X}$, and with edges $\{(x, y) \mid \exists c \in K.x \in c, y \in c\}$. Then each set $c \in K$ is

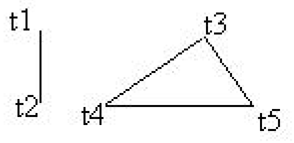

Figure 3: A Markov Network.

indeed a clique in the graph, justifying the terminology: in this paper we allow the cliques in K to be non-maximal, which is a minor departure from the standard definition. The importance of the graph is that edges correspond to correlations: an edge (x, y) means that the variables x, y are correlated, while the lack of an edge means that they are independent.

An MN is often represented as a *l*og-linear model, as follows: each potential function is given by $\Phi_c(W_c) = \texttt{exp}(w_c f_c(W_c))$, where $w_c \in \mathbb{R}$ is a *weight* and f_c is a *feature function*. Thus, the weight of a world is:

$$\Phi(W) = \texttt{exp}\left(\sum_{c \in K} w_c f_c(W \cap c)\right)$$

2.2 Size-Constrained Markov Networks (SCMN)

Our goal is to use a Markov Network to model soft keys. A clique will correspond to a set of tuples violating the key constraint, and therefore can be large: the basic assumption in Markov Networks that cliques are small no longer holds. On the other hand, to model soft keys, the clique's potential needs to be a function only of the number of tuples violating the key constraint, and not the actual set of tuples. For that purpose we introduce a restricted form of Markov Networks, which we call *Size Constrained* MN, and represent as a log-linear model:

DEFINITION 2.1. *A* Size-Constrained *Markov Network (SCMN) is* $\mathcal{M} = (\bar{X}, K, (w_i)_{i=1,n}, (f_c)_{c \in K})$*, where:*

- $\bar{X} = \{t_1, \ldots, t_n\}$ *is a set of Boolean variables. We refer to them as* tuples.
- K *is a set of nonempty subsets of* $\bar{X}$ *called* cliques.
- $w_i \in \mathbb{R}$ *is called the* weight *of the tuple* t_i.
- *For all* $c \in K$, $f_c : \{0, \ldots, |c|\} \rightarrow \mathbb{R}$*, s.t.* $f_c(0) = 0$.

An SCMN defines the following probability space on $\mathcal{W} = 2^{\bar{X}}$*:* $P(W) = \frac{1}{Z}\Phi(W)$*, where*

$$\Phi(W) = \textit{exp}\left(\sum_{t_i \in W} w_i + \sum_{c \in K} f_c(|W \cap c|)\right)$$
$$Z = \sum_{W \in \mathcal{W}} \Phi(W)$$

Thus, like in [15], the potential in an SCMN has two parts: one that depends only on the individual tuples, and one that depends only on the number of tuples in each cliques.

Example 2.2 We illustrate a Size-Constrained Markov Network over five Boolean variables $\bar{X} = \{t_1, \ldots, t_5\}$. The graph of the MN is in Fig. 3 and says that the variables t_1 and t_2 are correlated, and so are t_3, t_4, t_5, but that t_1, t_2 are independent from t_3, t_4, t_5. Define now the following SCMN:

$$\mathcal{M} = (\bar{X}, K, (w_i)_{i=1,5}, (f_c)_{c \in K})$$

where $w_1, \ldots, w_5 \in \mathbb{R}$ are the weights, $K = \{c_1, c_2\}$ and:

$$\begin{array}{lllll} c_1 & = & \{t_1, t_2\} & f_{c_1}(1) = v_1 \quad f_{c_1}(2) = v_2 & \\ c_2 & = & \{t_3, t_4, t_5\} & f_{c_2}(1) = u_1 \quad f_{c_2}(2) = u_2 & f_{c_2}(3) = u_3 \end{array}$$

Here $v_1, v_2, u_1, u_2, u_3 \in \mathbb{R}$. The probability function $P(W)$ multiplies the potentials e^{w_i} for all $t_i \in W$, then it examines the cardinalities of $W \cap c_1$ and of $W \cap c_2$ and multiplies with the corresponding potential. For example, for the two worlds $W = \{t_2, t_3, t_4\}$ and $W' = \{t_1, t_3, t_4, t_5\}$ we have:

$$P(W) = \frac{1}{Z}\texttt{exp}\,(w_2 + w_3 + w_4 + v_1 + u_2)$$
$$P(W') = \frac{1}{Z}\texttt{exp}\,(w_1 + w_3 + w_4 + w_5 + v_1 + u_3)$$

2.3 Soft Keys

We now formalize the notion of soft keys and describe their semantics using an SCMN.

Syntax We start by defining a probabilistic relational schema:

DEFINITION 2.3. *A* **probabilistic relation** *is a relation schema* $R(A_1, \ldots, A_k, W)$ *with a distinguished attribute* W*, called the* weight.

A probabilistic database schema is $\bar{R} = (R_1, \ldots, R_m)$, and a *probabilistic instance* is simply an instance I for $\bar{R}$: the term "probabilistic" refers to how we will interpret the weights, as we show below. As usual, we denote R_i^I the relation R_i of the instance I.

Given a relation name R we denote $Attr(R)$ the set of attributes without the weight attribute. We consider in this paper conjunctive queries that refer only to attributes in $Attr(R)$; thus, a *subgoal* g on R means a predicate (with variables and/or constants) over $Attr(R)$, without the weight attribute. We denote $vars(g)$ the set of variables in g. For example, give a relation $R(A, B, C, D, W)$, a subgoal is $g = R(x, a, y, x)$, where $vars(g) = \{x, y\}$.

DEFINITION 2.4. *Let R be a relation name.*

- *A* soft key schema *for R is a pair $\sigma = (\bar{x}, g)$ where $\bar{x}$ is a set of variables and g is a subgoal on R, s.t. $\bar{x} \subseteq vars(g)$. We denote $Key_\sigma(R) \subseteq Attr(R)$ the set of attributes in R that have in g either a constant or a $\bar{x}$-variable.*
- *A* soft key *for R is a triple $\gamma = (\sigma, s, w)$, where σ is a soft key schema, $s \in \mathbb{N}^+$ is called the* size *and $w \in \mathbb{R}$ is called the* weight.

We write $s = \texttt{size}(\gamma)$ and $w = \texttt{weight}(\gamma)$ to indicate the size and the weight of the soft key. We also use interchangeably σ and γ when clear from the context, e.g. write $Key_\gamma(R)$ instead of $Key_\sigma(R)$.

Informally, the soft key (σ, s, w) says this. If there exists s tuples in R with the same values $\bar{x}$, then charge with a weight w. If $w < 0$ then that will penalize s occurrences of x otherwise it will reward them.

Example 2.5 The first, second, and last soft key in Figure 2 are expressed as follows in our formalism:

$$((x, \texttt{Person}(x, y)), 2, -4)$$
$$((\emptyset, \texttt{Person}(\texttt{Frank}, y)), 2, 4)$$
$$((\emptyset, \texttt{Person}(a, y)), s, -2s)$$

The last line represents a set of soft keys: there is one for each a that satisfies $\texttt{Income}(a, z), z > 1M$ in the current database instance, and for each number s between 1 and the cardinality of `Person`. We will assume in this paper that the system performs automatically the conversion from a use-friendly syntax as in Fig. 2 to formal soft keys.

Semantics Given an instance I and a set of soft keys Γ, its semantics is given by the following SCMN:

$$\mathcal{M}(\Gamma, I) = (\bar{X}, K, (w_i)_{i=1,n}, (f_c)_{c\in K})$$

- $\bar{X} = \{t_1, \ldots, t_n\}$ is the set of tuples in all probabilistic relations in I. That is, $\bar{X} = \bigcup_{j=1,m} R_j^I$; we assume the union to be disjoint.
- $w_i = t_i.W$ (the weight of the tuple t_i in I).
- Let $\gamma \in \Gamma$ be a soft key for a relation R_j, i.e. $\gamma = (\sigma, s, w)$, where $\sigma = (\bar{x}, g)$, and denote $\bar{y} = vars(g) - \bar{x}$. Let $\bar{a}, \bar{b}$ be tuples of constants with the same arity as $\bar{x}$ and $\bar{y}$ respectively. Denote $g[\bar{a}/\bar{x}, \bar{b}/\bar{y}]$ the ground tuple obtained by substituting $\bar{x}$ with $\bar{a}$ and $\bar{y}$ with $\bar{b}$ in g. For any γ and $\bar{a}$, we define the following set:

  $$c_{\gamma,\bar{a}} = \{t_i \mid \exists\bar{b} : g[\bar{a}/\bar{x}, \bar{b}/\bar{y}] = t_i\}$$

  A set of the form $c_{\gamma,\bar{a}}$ consists of a subsets of tuples in R_j^I that are affected by the soft key γ. We define the set of cliques K to be all sets of this form, and the feature functions f_c to combine the weights of all soft keys that define the same clique c:

$$\begin{aligned} K &= \{c_{\gamma,\bar{a}} \mid c_{\gamma,\bar{a}} \neq \emptyset\} \\ f_c(s) &= \texttt{exp}\left(\sum_{\substack{\gamma\in\Gamma: \\ (\exists\bar{a}.c_{\gamma,\bar{a}} = c) \\ \wedge\texttt{size}(\gamma) = s}} \binom{|c|}{s}\texttt{weight}(\gamma)\right)\end{aligned}$$

We explain the definition through three examples. First, we examine a probabilistic instance I without any soft keys: in this case the probability defined by $\mathcal{M}(\emptyset, I)$ is precisely a tuple-independent probabilistic database [9].

Example 2.6 Let I be an instance of the probabilistic relational schema $R(A, B, W)$: $I = \{t_1, \ldots, t_n\}$. Denote $w_i = t_i.W$ the weight of tuple t_i in I, and let $p_i = e^{w_i}/(1 + e^{w_i})$. Consider the SCMN $\mathcal{M}(\emptyset, I)$; then the probability $P(W)$ of a possible world $W \subseteq I$ is (see Def 2.1):

$$P(W) = \frac{1}{Z}\texttt{exp}\left(\sum_{t_i\in W} w_i\right) = \frac{1}{Z}\prod_{t_i\in W}\frac{p_i}{1-p_i} \qquad (1)$$

From $\sum_W P(W) = 1$, through direct calculation we obtain $Z = 1/\prod_{i=1,n}(1-p_i)$. Thus, $P(W) = \prod_{t_i\in W} p_i \times \prod_{t_i\notin W}(1-p_i)$. This is precisely a tuple-independent probabilistic database: every tuple t_i appears in W independently, and its marginal probability is $P(t_i) = p_i$.

Now we examine how soft keys affect the probability distribution, by introducing correlations between tuples.

Example 2.7 Continuing the previous example, consider one soft key with schema:

$$\sigma = (x, R(x, y))$$

For concreteness, suppose the soft key has size 3 and weight -5.0, i.e. $\gamma = (\sigma, 3, -5.0)$. This says that three or more occurrences of the same value for A should be penalized by -5.0. Suppose a world W contains three tuples $(a, b_1), (a, b_2), (a, b_3)$: then the SCMN semantics penalizes $P(W)$ by multiplying Eq.(1) by e^{-5}. Now suppose that W contains n violations, $(a, b_1), \ldots, (a, b_n)$: then Eq.(1) is multiplied with $e^{-5\binom{n}{3}}$: every three distinct occurrences of a contribute with a weight of -5.0. This is a desirable behavior: the user says that three occurrences should be penalized by -5.0, and therefore she expects the penalty to increase with n. Our particular choice of defining this penalty, by multiplying with $\binom{n}{3}$, is somewhat arbitrary. Our choice was inspired by Markov Logic [22]: this is precisely the semantics obtained in Markov Logic if one assigns weight -5.0 to the formula:

$$\exists y_1.\exists y_2.\exists y_3. \bigwedge_i R(x, y_i) \wedge \bigwedge_{i\neq j} y_i \neq y_j$$

However, other choices are possible, for example one could multiply with $n - 3$ instead of $\binom{n}{3}$. The results in this paper are not affected by the particular choice of the feature functions f_c.

Clearly, by adding the soft key γ, we introduced correlations between tuples that share the same value of A. But, in addition, the soft key changes the marginal probability of *every* tuple $t_i = R(a, b)$, even if it does not violate the soft key, because Z changes. In example 2.6 the marginal probability of a tuple t_i was simply $P(t_i) = p_i$. By adding a single soft key constraint it becomes unclear how to compute the marginal probability of any tuple, even one that doesn't violate the key constraint.

Finally, we examine the impact of several keys.

Example 2.8 We add a second key to the previous example:

$$\gamma' = (\emptyset, R(x, y), 1000, +3.5)$$

Now $\Gamma = \{\gamma, \gamma'\}$. Here γ' rewards worlds that have at least 1000 tuples. In other words, while γ encourages us to remove tuples that share the same key, γ' penalizes us if we remove too many globally: more precisely by rewarding worlds with over 1000 tuples it increases Z and thus penalizes worlds with less than 1000 tuples. This illustrates that multiple soft keys may interact and further complicate the probability space: with this new soft key it seems even more difficult to compute the marginal probability of a tuple.

PROPOSITION 2.9. *The size of* $\mathcal{M}(\Gamma, I)$ *is bounded by a polynomial in the size of* Γ *and* I.

PROOF. (Sketch) Let $n = |I|$ and $m = |\Gamma|$. Consider a soft key $\gamma \in \Gamma$: the cliques it generates, $c_{\gamma,\bar{a}}$, are disjoint sets, hence γ generates at most n cliques. Thus there are at most mn cliques in $\mathcal{M}(\Gamma, I)$. $\Box$

2.4 Problem Definition

We fix the probabilistic relational schema $\bar{R}$ and a Boolean conjunctive query q, and study the following problem: given a set of soft keys Γ and an instance I for $\bar{R}$, compute the marginal probability $P(q)$:

$$P(q) = \sum_{W\subseteq I:W\models q} P(W) \qquad (2)$$

where P is the probability distribution defined by $\mathcal{M}(\Gamma, I)$. We call $P(q)$ the *value of* q *on* I *given the soft keys* Γ.

Notice that the soft keys Γ are part of the input; this is because we insist on evaluating q using an algorithm that is generic in the sizes and weights in Γ.

In this paper we restrict our discussion to conjunctive queries without self-joins, i.e. where every relation name occurs at most once in the query. For example the two queries $R(x, y), S(y, a, z)$ and $R(a, x, x), S(x, y, y, b), T(x, z)$ are without self-joins, while the query $R(x, y), R(y, z)$ has a self-join. This restriction is similar to other complexity results for the query evaluation on probabilistic databases [10, 20, 9, 21]. Queries that have self-joins are significantly harder to analyze: the only case that has been studied is that of queries over tuple-independent databases [8], and that turned out to be significantly harder.

3. SINGLE SUBGOAL QUERIES

We start our investigation by studying the complexity of queries consisting of a single subgoal. Examples include $q : -R(x, a, y)$, or $q : -R(x, x, x)$ or $q : -R(x, y, z)$. These are select-project queries, and we call them in this section *disjunctive* queries. In the presence of soft keys, even such queries can be hard:

PROPOSITION 3.1. *Consider a relation $R(A, B, W)$ and the query $q : -R(x, y)$ (which simply checks if R is non-empty). Consider two key schemas: $\sigma_1 = (x, R(x, y))$ and $\sigma_2 = (y, R(x, y))$. Then the problem: for inputs I, w_1, w_2, compute the value of q on I given the soft keys $(\sigma_1, 2, w_1)$ and $(\sigma_2, 2, w_2)$ is $\#P$-hard.*

PROOF. We will first prove hardness for $w_1 = w_2 = -\infty$, then show how to extend the proof to $w_1, w_2 \in \mathbb{R}$. We use a reduction from the IMPERFECT MATCHING (IPM) problem, which is: given a bipartite graph $G = (U, V, E)$, with $E \subseteq U \times V$, compute the number of matches (full or partial). A match is a $M \subseteq E$ s.t. every $u \in U$ occurs at most once, and every $v \in V$ occurs at most once in M. IPM is #P-hard as shown in [28]. Given a graph $G = (U, V, E)$ for the IPM problem, we reduce it to an instance I over the schema $R(A, B, W)$ as follows: $I = \{(u, v, 0) | (u, v) \in E\}$. Thus, each tuple has weight 0. A world $W \in \mathcal{W}$ corresponds to a subset of edges $M \subseteq E$. Denote $\mathcal{W}_1$ the set of worlds that are matches, and $\mathcal{W}_0 = \mathcal{W} - \mathcal{W}_1$. If $W \in \mathcal{W}_1$ then $\Phi(W) = 1$, and if $W \in \mathcal{W}_0$ then $\Phi(W) = 0$ (because either γ_1 or γ_2 are violated and their weights are $-\infty$). Denote $m = |\mathcal{W}_1|$ the number of partial matches in G. We have:

$$\begin{aligned} Z &= \sum_{W \in \mathcal{W}} \Phi(W) = m \\ \Phi(q) &= \sum_{W \in \mathcal{W}, W \models q} \Phi(W) = Z - 1 \end{aligned}$$

because the only world in $\mathcal{W}$ that does not satisfy q is $\emptyset$, which is also a partial match. It follows that $P(q) = \Phi(q)/Z = (m-1)/m$, hence $m = 1/(1 - P(q))$. This completes the reduction, and shows that computing $P(q)$ is #P-hard when $w_1 = w_2 = -\infty$. Now we prove hardness assuming that w_1, w_2 are in $\mathbb{R}$, and part of the input. Define $w_1 = w_2 = w$; we choose w later. Denoting $\varepsilon = \sum_{W \in \mathcal{W}_0} \Phi(W)$ we have $Z = m + \varepsilon$ and $\Phi(q) = Z - 1$, hence $m = 1/(1 - P(q)) - \varepsilon$. We choose w small enough to ensure $\varepsilon \leq 1/2$: this allows us to compute m as $\lfloor 1/(1 - P(q)) \rfloor$. Namely, choose w s.t. $e^w \leq 1/2^{|E|+1}$, thus $\forall W \in \mathcal{W}_0$, $\Phi(W) \leq 1/2^{|E|+1}$, which implies $\varepsilon \leq 1/2$ because there are only $2^{|E|}$ possible worlds. $\square$

Thus, the interaction between soft keys on the same relation can make even a disjunctive query hard. We show, however, that if the soft keys are "hierarchical", then any disjunctive query can be computed in polynomial time.

DEFINITION 3.2. *Let γ_1, γ_2 be two soft keys on a relation R. We say that they are* hierarchical *if either (1) $Key_{\gamma_1}(R) \subseteq Key_{\gamma_2}(R)$ or (2) $Key_{\gamma_1}(R) \supseteq Key_{\gamma_2}(R)$ or (3) the subgoals g_1, g_2 of γ_1 and γ_2 are incompatible (i.e. non-unifiable): we write $g_1 \cap g_2 = \emptyset$ to indicate that g_1, g_2 are incompatible.*

Let Γ be a set of soft keys on R. We say that Γ is hierarchical if $\forall \gamma_1, \gamma_2 \in \Gamma$, the pair γ_1, γ_2 is hierarchical.

For example, consider the following soft keys on R(A,B,C,W) (we show only the key schemas: the sizes and weights are not used in the definition):

$$\begin{aligned} \gamma_1 &= (y_1, R(x_1, a_1, y_1)) \\ \gamma_2 &= (y_2, R(y_2, a_2, x_2)) \\ \gamma_3 &= ((y_3, y_4), R(x_3, y_3, y_4)) \end{aligned}$$

- γ_1, γ_2 are hierarchical (assuming $a_1 \neq a_2$). This is because $R(x_1, a_1, y_1)$ and $R(y_2, a_2, x_2)$ are incompatible: in our notation, $R(x_1, a_1, y_1) \cap R(y_2, a_2, x_2) = \emptyset$.
- γ_2, γ_3 are non-hierarchical: $Key_{\gamma_2}(R) = \{A, B\}$, $Key_{\gamma_3}(R) = \{B, C\}$, and their subgoals are compatible.
- γ_1, γ_3 are hierarchical because $Key_{\gamma_3}(R) = \{B, C\}$ and $Key_{\gamma_1}(R) = \{B, C\}$.

We prove two results in this section:

THEOREM 3.3. *If Γ is hierarchical, then any disjunctive query on R can be evaluated in time $O(n^{1+arity(R)})$.*

THEOREM 3.4. *If Γ is non-hierarchical and contains no constants, then every disjunctive query on R is #P-hard.*

3.1 A PTIME Algorithm for Hierarchical SCMN

We prove here Theorem 3.3. In fact, we will prove a more general result. Fix any SCMN $\mathcal{M} = (\bar{X}, K, (w_i)_{i=1,n}, (f_c)_{c \in K})$, and assume w.l.o.g. that $\forall c \in K$, $|c| \geq 2$.

DEFINITION 3.5. *$\mathcal{M}$ is* hierarchical *if $\forall c_1, c_2 \in K$ either $c_1 \cap c_2 = \emptyset$ or $c_1 \subseteq c_2$ or $c_1 \supseteq c_2$. The* height *of the hierarchy is the largest number h s.t. there exists h cliques in K s.t. $c_1 \supset c_2 \supset \ldots \supset c_h \neq \emptyset$.*

THEOREM 3.6. *Let $\mathcal{M}$ be hierarchical SCMN of height h and $Q \subseteq \bar{X}$ a set of tuples. Then the probability $P(Q)$ defined as:*

$$P(Q) = \sum_{W \subseteq \mathcal{W} : Q \cap W \neq \emptyset} P(W)$$

can be computed in time $O(n^{h+1})$, where $n = |\bar{X}|$.

Theorem 3.6 proves Theorem 3.3: this is because if Γ is hierarchical then for every I the SCMN $\mathcal{M}(\Gamma, I)$ is hierarchical and its height is $\leq arity(R)$. Moreover, on the probabilistic instance I, the query q is equivalent to a fixed set of tuples Q. More precisely, defining $Q \subseteq I$ (note that here $\bar{X} = I$) to be the set of tuples that match the subgoal q, then we have for every world $W \subseteq I$: $W \models q$ iff $W \cap Q \neq \emptyset$.

In the remainder of this section we prove Theorem 3.6.

The proof of the theorem is given by the Algorithm 3.1, which computes $P(Q)$ using dynamic programming. We first describe the notations used by the algorithm and explain it, then prove its correctness and running time.

Algorithm 3.1 Computing $P(Q)$ on a hierarchical SCMN

```
1:  Inputs: SCMN $(\bar{X}, K, (w_i)_{i=1,n}, (f_c)_{c\in K})$, query $Q \subseteq \bar{X}$.
2:  Outputs: $P(Q)$.
3:  Let $Z^0_\varepsilon = 1$ $S^0_\varepsilon = 0$
4:  for $k = 1, n$ do
5:    for $s_1, \ldots, s_{h_k} \in \{0, \ldots, k\}^{h_k}$ do
6:      Let $\bar{s} = (s_1, \ldots, s_{h_k})$
7:      Let $c_1 \supset \ldots \supset c_L$ the common ancestors of $k-1, k$
8:      if $\bigwedge_{i=L+1}^{h_k}(s_i = 0)$ then
9:        Let $U = \sum_{\bar{s}':\bar{s}'_L=\bar{s}_L} Z^{k-1}_{\bar{s}'}$ and $R = \sum_{\bar{s}':\bar{s}'_L=\bar{s}_L} S^{k-1}_{\bar{s}'}$
10:     else
11:       $U = R = 0$
12:     end if
13:     if $\bigwedge_{i=L+1}^{h_k}(s_i = 1)$ then
14:       Let $F = \mathtt{exp}(w_k + \sum_{i=1,L}(f_{c_i}(s_i) - f_{c_i}(s_i - 1)))$
15:       $V = F \sum_{\bar{s}':\bar{s}'_L=\bar{s}_L-1} Z^{k-1}_{\bar{s}'}$
16:       if $t_k \in Q$ then
17:         $T = V$
18:       else
19:         $T = F \sum_{\bar{s}':\bar{s}'_L=\bar{s}_L-1} S^{k-1}_{\bar{s}'}$
20:       end if
21:     else
22:       Let $T = V = 0$
23:     end if
24:     Let $Z^k_{\bar{s}} = U + V$ and $S^k_{\bar{s}} = R + T$
25:   end for
26: end for
27: Let $\Phi(Q) = \sum_{\bar{s}} S^n_{\bar{s}}$ and $Z = \sum_{\bar{s}} Z^n_{\bar{s}}$
28: Return $P(Q) = \Phi(Q)/Z$.
```

We start by defining the following forest T:

$$\begin{aligned} \mathtt{Nodes}(T) &= K \cup \{\{t_i\} \mid t_i \in \bar{X}\} \\ \mathtt{Edges}(T) &= \{(c, c') \mid c \supset c' \wedge \neg\exists c''.(c \supset c'' \supset c')\} \end{aligned}$$

The leaves of this forest correspond precisely to the tuples $t_1, \ldots, t_n$. Fix any order on the forest: this defines both an order on the leaf nodes,$t_1, t_2, \ldots, t_n$, and of the internal nodes. The order ensures that all tuples belonging to a clique form a subsequence of the leaf nodes: $t_i, t_{i+1}, \ldots, t_j$. This was possible because the cliques are hierarchical.

We now define $O((n+1)^{h+1})$ subsets of $\mathcal{W}$. For $k = \{0, \ldots, n\}$, denote:

- $\bar{X}_k = \{t_1, \ldots, t_k\}$ (the first k tuples).
- $h_k =$ the number of cliques containing t_k (for $k = 0$, we set $h_k = 0$). Note that $h_k \leq h$.
- $c_1 \supset c_2 \supset \ldots \supset c_{h_k} \supset \{t_k\}$ the longest path in T to t_k.

DEFINITION 3.7. *Let $k \in \{0, \ldots, n\}$ and $\bar{s} = (s_1, \ldots, s_{h_k})$, where $s_1, \ldots, s_{h_k} \in \{0, \ldots, n\}$. Define:*

$$\begin{aligned} \mathcal{W}^k_{\bar{s}} &= \{W \mid W \subseteq \bar{X}_k \wedge \forall i \in [h_k].|W \cap c_i| = s_i\} \\ \mathcal{Q}^k_{\bar{s}} &= \{W \mid W \in \mathcal{W}^k_{\bar{s}} \wedge W \cap Q \neq \emptyset\} \end{aligned}$$

In other words, $\mathcal{W}^k_{\bar{s}}$ consists of all worlds that (a) use only the first k tuples, and (b) their intersection with the cliques $c_1, \ldots, c_{h_k}$ on the path from a root to t_k have cardinalities $s_1, \ldots, s_{h_k}$. Note that we have the following:

$$\begin{aligned} W^0_\varepsilon &= \{\emptyset\} \\ \mathcal{W} &= \bigcup_{\bar{s}} W^n_{\bar{s}} \end{aligned}$$

The algorithm computes iteratively $O((n+1)^{h+1})$ quantities:

$$Z^k_{\bar{s}} = \sum_{W \in \mathcal{W}^k_{\bar{s}}} \Phi(W) \qquad S^k_{\bar{s}} = \sum_{W \in \mathcal{Q}^k_{\bar{s}}} \Phi(W)$$

Then $Z = \sum_{\bar{s}} Z^n_{\bar{S}}$, and $\Phi(Q) = \sum_{\bar{s}} S^n_{\bar{s}}$, which allows us to compute $P(Q) = \Phi(Q)/Z$. We still need to introduce a few notation used by the algorithm. Given $\bar{s} = (s_1, s_2, \ldots, s_d)$ denote:

$$\begin{aligned} \bar{s} - 1 &= (s_1 - 1, s_2 - 1, \ldots, s_d - 1) \\ \bar{s}_L &= (s_1, s_2, \ldots, s_L) \text{ for } L \leq d \end{aligned}$$

Given two consecutive leaves t_{k-1}, t_k in the forest T, we denote with $c_1 \supset c_2 \supset \ldots \supset c_L$ all their common ancestors.

We now prove the correctness of the algorithm.

Consider a world $W \in \mathcal{W}^k_{\bar{s}}$. There are two cases. The first case is when $t_k \notin W$; then W contains at most the tuples $t_1, \ldots, t_{k-1}$. Recall that $c_1 \supset \ldots \supset c_L$ are the common ancestors of t_{k-1} and t_k, thus $W \in \mathcal{W}^{k-1}_{\bar{s}'}$ for some $\bar{s}'$ s.t. $s'_1 = s_1, \ldots, s'_L = s_L$. Let $c_{L+1} \supset \ldots \supset c_{h_k}$ be the rest of the path to t_k: in all these cliques, t_k is the smallest element, hence W cannot contain any tuples from these cliques. We therefor must have $s_{L+1} = \ldots = s_{h_k} = 0$. This completes the analysis of the first case. The second case is when $t_k \in W$: then we must have $W - \{t_k\} \in \mathcal{W}^{k-1}_{\bar{s}'}$, where $s'_1 = s_1 - 1, \ldots, s'_L = s_L - 1$ (since we removed t_k), we can argue similarly that $s_{L+1} = \ldots = s_{h_k} = 1$. This allows us to derive a recursive formula for $\mathcal{W}^k_{\bar{s}}$. We can derive a similar one for $\mathcal{Q}^k_{\bar{s}}$: the only extra wrinkle here is that, when $t_k \in W$ then we also need to check if $t_k \in Q$: if not then we recur with $\mathcal{Q}^k_{\bar{s}'}$; if yes, then we recur with $\mathcal{Z}^k_{\bar{s}'}$. We state the resulting recurrence formally. In the lemma below we denote $\mathcal{A} \cup \{t\}$ the set $\{W \cup \{t\} \mid W \in \mathcal{A}\}$ for a set of worlds $\mathcal{A} \subseteq \mathcal{W}$ and a tuple t:

LEMMA 3.8. *$\mathcal{W}^k_{\bar{s}} = \mathcal{U}^k_{\bar{s}} \cup \mathcal{V}^k_{\bar{s}}$, where*

$$\mathcal{U}^k_{\bar{s}} = \begin{cases} \bigcup_{\bar{s}':\bar{s}'_L=\bar{s}_L} \mathcal{W}^{k-1}_{\bar{s}'} & \text{if } \bigwedge_{i=L+1}^{h_k} s_i = 0 \\ \emptyset & \text{otherwise} \end{cases}$$

$$\mathcal{V}^k_{\bar{s}} = \begin{cases} \bigcup_{\bar{s}':\bar{s}'_L=\bar{s}_L-1} \mathcal{W}^{k-1}_{\bar{s}'} \cup \{t_k\} & \text{if } \bigwedge_{i=L+1}^{h_k} s_i = 1 \\ \emptyset & \text{otherwise} \end{cases}$$

If $t_k \in Q$ then $\mathcal{Q}^k_{\bar{s}} = \mathcal{R}^k_{\bar{s}} \cup \mathcal{V}^k_{\bar{s}}$ and if $t_k \notin Q$ then $\mathcal{Q}^k_{\bar{s}} = \mathcal{R}^k_{\bar{s}} \cup \mathcal{T}^k_{\bar{s}}$, where:

$$\mathcal{R}^k_{\bar{s}} = \begin{cases} \bigcup_{\bar{s}':\bar{s}'_L=\bar{s}_L} \mathcal{Q}^{k-1}_{\bar{s}'} & \text{if } \bigwedge_{i=L+1}^{h_k} s_i = 0 \\ \emptyset & \text{otherwise} \end{cases}$$

$$\mathcal{T}^k_{\bar{s}} = \begin{cases} \bigcup_{\bar{s}':\bar{s}'_L=\bar{s}_L-1} \mathcal{Q}^{k-1}_{\bar{s}'} \cup \{t_k\} & \text{if } \bigwedge_{i=L+1}^{h_k} s_i = 1 \\ \emptyset & \text{otherwise} \end{cases}$$

We can now prove:

PROPOSITION 3.9. *Algorithm 3.1 correctly computes the probability $P(Q)$ of a disjunctive query Q over a hierarchical SCMN.*

PROOF. We need to show that the quantities $Z^k_{\bar{s}}$ and $S^k_{\bar{s}}$ are computed correctly. This follows immediately from the previous lemma, observing that the values denoted U, V, R, T in the algorithm are precisely $\Phi(\mathcal{U}^k_{\bar{s}})$, $\Phi(\mathcal{V}^k_{\bar{s}})$, $\Phi(\mathcal{R}^k_{\bar{s}})$, $\Phi(\mathcal{T}^k_{\bar{s}})$ The crux of the correctness proof relies in examining the case $t_k \in W$: then $\Phi(W) = \Phi(W - \{t_k\})F$, where the factor F is $\texttt{exp}(w_k + \sum_{i=1,L} f_{c_i}(s_i) - f_{c_i}(s_i - 1))$: this is because W contributes in addition to $W - \{t_k\}$ with the weight w_k for the tuple t_k and with the weight $f_{c_i}(s_i)$ for the clique c_i: on the other hand $W - \{t_k\}$ contributes with a weight $f_{c_i}(s_i - 1)$ for that clique, which justifies the formula for F. $\square$

3.2 Hardness of Non-hierarchical Keys

In this section we prove Theorem 3.4. For that we first extend Proposition 3.1:

PROPOSITION 3.10. *Consider the relation $R(A, B, W)$ and the key schemas $\sigma_1 = (x, R(x, y)$ and $\sigma_2 = (y, R(x, y))$ as in Proposition 3.1. Consider the following four queries:*

$$\begin{aligned} q_1 &:- & R(x, y) \\ q_2 &:- & R(x, x) \\ q_3 &:- & R(a, y) \\ q_4 &:- & R(a, b) \end{aligned}$$

where a, b are constants. Then, for each of the queries q_i, $i = 1, 2, 3, 4$, the problem: for inputs I, w_1, w_2, compute the value of q_i on I given the soft keys $(\sigma_1, 2, w_1)$ and $(\sigma_2, 2, w_2)$ is #P-hard.

Note that this extends Proposition 3.1 from q_1 to q_2, q_3, and q_4. We refer the reader to the full version of this paper for the proof.

Now we can prove Theorem 3.4 by reduction from one of the three queries in Proposition 3.10. Let Γ be a set of soft keys without constants. Since it is non-hierarchical there exists two soft keys γ_1, γ_2 s.t. $Key_{\gamma_1}(R)$ and $Key_{\gamma_2}(R)$ are incomparable sets. (Note that case (3) of Definition 3.2 cannot happen because there are no constants). Thus, there exists two attributes A, B s.t. $A \in Key_{\gamma_1}(R) - Key_{\gamma_2}(R)$ and $B \in Key_{\gamma_2}(R) - Key_{\gamma_1}(R)$. We examine now the query q on these two attributes (recall that q is a single subgoal): it can have two distinct variables, the same variable, a variable and a constant, or two constants. We then do a reduction from the corresponding query q_i in Proposition 3.10, by setting the weights of all soft keys other than γ_1, γ_2 to 0.

4. CONJUNCTIVE QUERIES

Consider a relational schema $\bar{R}$ with probabilistic relations R_1, $\ldots, R_m$. Let $\Gamma = \{\gamma_1 \cup \ldots \cup \gamma_m\}$ be the set of soft keys s.t. the soft key γ_i applies to R_i. Thus, each relation R_i has exactly one soft key: this includes the case when there is no soft key for R_i, because in that case we can define the soft key for R_i to be the no-op key, whose schema is $(\bar{x}, R_i(\bar{x}))$, where $R_i(\bar{x})$ has a distinct variable for each attribute: $R_i(\bar{x}) = R_i(x_1, x_2, \ldots)$. In this section we impose a restriction on the soft keys. Their subgoals may have no repeated variables: e.g. we allow $((x, y), R(y, a, z, b, x))$ but not $(x, R(x, x, y, y))$. Also assume there are no trivial soft keys i.e. they have non-zero weight otherwise it doesn't make sense to consider them.

DEFINITION 4.1. *Let g be a subgoal over R whose soft key has schema $\gamma = (\bar{x}, g')$. We define the set $Key(g) \subseteq Vars(g)$, as follows. First, if $g \cap g' = \emptyset$ (see Def. 3.2) then $Key(g) = Vars(g)$; otherwise $Key(g)$ consists of all variables in g that occur on a position where g' has a key variable.*

For example, consider the relation $R(A, B, C, W)$ with soft key $\gamma = (x, R(a, x, y))$. Then

$$\begin{aligned} Key(R(a, y, z)) &= \{y\} \\ Key(R(b, y, z)) &= \{y, z\} \\ Key(R(x, b, z)) &= \emptyset \end{aligned}$$

Let q be a conjunctive query. For a variable $x \in Vars(q)$ we denote $Sg(x) = \{g \mid x \in Key(g)\}$. Thus, $Sg(x)$ contains all subgoals in which x occurs in a key position.

DEFINITION 4.2 (SAFE QUERIES). *Let q be a boolean conjunctive query without self-joins. q is* safe *for soft keys Γ if one of the following holds:*

1. **Base case:** *$q := g$, where g is a single subgoal.*
2. **Disconnected components:** *$q = q_1 q_2$ where $Vars(q_1) \cap Vars(q_2) = \varnothing$, and q_1 and q_2 are both safe.*
3. **Projectable Variable:** *$\exists x \in Vars(q)$, such that (a) $\forall g \in Sg(q), x \in Vars(g)$ (i.e. x appears in all subgoals), (b) $\forall y \in Vars(q)$, $Sg(y) \subseteq Sg(x)$ and (c) for every constant a, $q[a/x]$ is safe.*

Thus, a variable x is a *projectable variable* then x must occur in all subgoals. It does not have to occur everywhere in key positions, but the set of subgoals where it does not occur in a key position should not have any other key variable either. Note that the safety for $q[a/x]$ is independent of the choice of the constant a.

Example 4.3 We illustrate with two queries, and underline in each subgoal g the variables in $Key(g)$: thus, $R(\underline{x}, y)$ means that there exists a soft key with schema $(x, R(x, y))$, while $S(y)$ means there exists a soft key $(\emptyset, S(y))$:

$$\begin{aligned} q_1 &= R(\underline{x}, y), S(x) \\ q_2 &= R(\underline{x}, y), S(y) \end{aligned}$$

q_1 is safe: x occurs everywhere, and $Sg(x) = \{R\}$ while $Sg(y) = \emptyset$. On the other hand q_2 is unsafe: x does not occur everywhere. While y does occur everywhere, $Sg(y) = \emptyset$, while $Sg(x) = \{R\}$.

Intuitively, *projectable variable* condition states that there is a variable x in all the sub-goals of q and all other variables y in each subgoal g are such that $y \leq_g x$. We call x the projectable variable.

THEOREM 4.4. *Let Γ be a set of soft keys, with one key per relation. Let q be a conjunctive query q without self-joins. If q is safe for Γ then it can be evaluated in PTIME. If the query is unsafe for Γ, then it is #P-hard.*

In the remainder of this section we prove the theorem.

4.1 Hardness of unsafe queries

We start by proving that every unsafe query is #P-hard. We use a result in [9] that establishes a dichotomy of conjunctive queries without self-joins on disjoint-independent probabilistic databases. We briefly review that result, using the terminology in our paper.

DEFINITION 4.5. *A soft key $\gamma = (\sigma, s, w)$ on a relation R is called a* hard key *if $s = 2$ and $w = -\infty$.*

A hard key is called a standard key *if its subgoal g has no constants.*

A standard key is called a trivial key *if $\bar{x} = Vars(g)$.*

We illustrate with the three keys on $R(A, B, C, W)$:

$$\begin{aligned}\gamma_1 &= ((x, R(x, a, y)), 2, -\infty) \\ \gamma_2 &= ((x, R(x, y, z)), 2, -\infty) \\ \gamma_3 &= (((x, y, z), R(x, y, z)), 2, -\infty)\end{aligned}$$

γ_1 is hard, γ_2 is standard (it says that A is a key), and γ_3 is trivial (it says that A, B, C are a key, which is the same as not giving any key for R).

DEFINITION 4.6. *A disjoint-independent probabilistic database instance is an instance I together with a set of standard keys Γ.*

We now review the definition of safe queries over disjoint-independent probabilistic databases, which we call here *h-safe* queries to distinguish them from ours, and review the dichotomy result on disjoint-independent probabilistic databases. Recall that $Key(g)$ denotes the set of variables that appear in a key position; denote $NKey(g)$ the set of variables that appear in a non-key position in g. These sets are not necessarily disjoint, because variables may be repeated.

DEFINITION 4.7. *[9] A Boolean conjunctive query is* h-safe *for a set of keys Γ if:*

1. **Base case** *$q := g$ where g is a single subgoal.*
2. **Disconnected components** *$q = q_1 q_2$ where $Vars(q_1) \cap Vars(q_2) = \emptyset$ and q_1, q_2 are both safe.*
3. **Independent project** *$\exists x \in Vars(q)$ s.t. $\forall g \in Sg(q)$, $x \in Key(g)$, and for any constant a, $q[a/x]$ is safe. Thus, x must occur in a key position in every subgoal.*
4. **Disjoint project** *$\exists g \in Sg(q)$ s.t. $Key(g) = \emptyset$ and $\exists y \in NKey(g)$ s.t. $q[a/y]$ is safe. Thus, y must occur in a non-key position in g and g has no key variables.*

The dichotomy for disjoint-independent databases is:

THEOREM 4.8. *[9] Let Γ be a set of hard keys. If a query q is h-safe for Γ, then it can be computed in PTIME. If it is not h-safe, then it is #P-hard.*

Now consider a set of soft keys Γ without variables for the relations $\bar{R}$. A *hardening* of Γ is a set of keys Γ^h obtained by either hardening or trivializing every soft key in Γ: that is, for every key (σ, s, w) in Γ there is a key $(\sigma, 2, w)$ in Γ^h where $w = -\infty$ or $w = 0$. We prove the following:

PROPOSITION 4.9. *Let Γ be a set of soft keys without variables and q be a query. Then q is safe w.r.t. Γ iff for every hardening Γ^h of Γ q is h-safe w.r.t. Γ^h.*

PROOF. We prove "only if" by induction on the structure of the query. The interesting case is given by condition 3: the others are straightforward. Let x occur in all subgoals. If x occurs in each subgoal in a key position, then the *independent project* case applies and we conclude that q is h-safe. So suppose x occurs in some subgoal g only on non-key positions. Then no variable y can occur in g on a key position, otherwise $sg(y) \not\subseteq sg(x)$. So $Key(g) = \emptyset$, hence x appears in a non-key position and we apply a disjoint project on x, hence q is h-safe.

We now prove the "if" direction. Here also we consider condition 3 of the safety definition, and assume that it fails. Then we "harden" the key constraints as follows. Let $G_1 = \{g \mid Key(g) = \emptyset\}$ and $G_2 = \{g \mid Key(g) \neq \emptyset\}$. For all $g \in G_1$ we trivialize its key, hence $Key^h(g) = Vars(g)$. For all $g \in G_2$ we harden the key, hence $Key^h(g) = Key(g)$. We prove that the resulting query is h-unsafe. Indeed, a disjoint project is not possible, since we trivialized all keys in G_1 where such a project would have been possible. Suppose an independent project is possible on some variable x. We show that x is a projectable variable by checking condition 3. Clearly it occurs in all subgoals. In addition $sg(x) = G_2$. Moreover, for any variable y, $sg(y) \subseteq G_2$, since no variables occur in key positions in G_1. Contradiction. $\Box$

Recall the query $q_1 = R(\underline{x}, y), S(x)$ in Example 4.3. There are four ways to harden it: in each subgoal g either keep $Key(g)$ unchanged, or include all variables. Each of these is h-safe. For example q_1 is h-safe because we do a disjoint project on y; $R(\underline{x}, y), S(\underline{x})$ is h-safe because we do an independent project on x. On the other hand q_2 is not safe, because the hardening $R(\underline{x}, y), S(\underline{y})$ is h-unsafe.

We use this to prove:

COROLLARY 4.10. *If a query q is unsafe for a set of soft keys Γ then q is #P-hard.*

For the proof, we first use Prop.4.9 to harden the keys s.t. q is h-unsafe w.r.t. Γ. Next, we observe that we can remove the constants in these hard keys by eliminating the corresponding attribute in the relation: the new query (over a new schema) is still h-unsafe, but now all keys are standard: thus the new query is #P-hard by Theorem 4.8. Finally, use the same technique as in Proposition 3.1 to replace weights $-\infty$ with weights $w \in \mathbb{R}$.

4.2 Algorithm for safe queries

We explain and prove the algorithm with the following sequence of definitions and proofs.

Given a soft key $\gamma = (\_, g)$ we denote g by g_γ.

DEFINITION 4.11. **Homogenization :** *Consider a relation R which has a soft key γ. Given a subgoal g over R, the homogenized instance $\texttt{hom}(R, g, \gamma)$ is defined as $R^t = \sigma_{\bar{A}=\bar{a}}R$, where $\bar{A}$ are the attribute positions where g_γ has key attributes and g has constants $\bar{a}$.*

Example 4.12 $\texttt{hom}(R(A, B, C), R(a, y, z), (x, y, R(x, y, z))) = \sigma_{A=a}R$
$\texttt{hom}(R(A, B, C), R(x, y, a), (x, y, R(x, y, z))) = R$
$\texttt{hom}(R(A, B, C), R(a, b, a), (y, R(a, y, z))) = \sigma_{A=a,B=b}R$

Consider a CQ $q = g_1 \dots g_m$ over $\bar{R}$. Since the query has no self join we can assume g_i is over R_i with soft key $\Gamma_i = (\_, G_i)$. We assume that the subgoals of the query have no repeated variables. For e.g. $R(x, x, y)$ is not allowed. This is just for the ease of presentation; the algorithm can be made to work for that case as well.

DEFINITION 4.13. *Let $\bar{s} = (s_1, \dots, s_m)$ with $s_i \in \{\mathbb{N} \cup \{0\}\}$. Define:*

$$\begin{aligned}\mathcal{W}_{\bar{R},\bar{s}} &= \{W \mid W \subseteq \bar{R} \wedge \forall i |W \cap R_i \cap G_i| = s_i\} \\ \mathcal{Q}_{\bar{R},\bar{s}} &= \{W \mid W \in \mathcal{W}_{\bar{R},\bar{s}} \wedge W \models q\} \\ Z_{\bar{R},\bar{s}} &= \sum_{W \in \mathcal{W}_{\bar{R},\bar{s}}} \Phi(W) \\ \Phi_{\bar{R},\bar{s}} &= \sum_{W \in \mathcal{Q}_{\bar{R},\bar{s}}} \Phi(W)\end{aligned}$$

So $\mathcal{W}_{\bar{R},\bar{s}}$ is the set of all worlds of $\bar{R}$ s.t. $\forall 1 \leq i \leq m$,the number of tuples from R_i that are influenced by the soft key are exactly s_i. $\mathcal{Q}_{\bar{R},\bar{s}}$ is the subset of $\mathcal{W}_{\bar{R},\bar{s}}$, where query is true. $Z_{\bar{R},\bar{s}}$

Algorithm 4.1 To compute $\Phi_{\bar{R},\bar{s}}(q)$, $Z_{\bar{R},\bar{s}}$

```
1:  **Inputs:** CQ $q$, over $\bar{R}$. $\bar{s} \in \{\mathbb{N} \cup 0\}^m$.
2:  **Outputs:** $\Phi_{\bar{R},\bar{s}}(q)$, $Z_{\bar{R},\bar{s}}(q)$
3:  if $q = q_1 q_2$ and $Vars(q_1) \cap Vars(q_2) = \emptyset$ then
4:      Calculate the components corresponding to $q_1$ and $q_2$ separately and then multiply them
5:  end if
6:  if $q$ is a single subgoal query then
7:      Use Algorithm 3.1
8:  end if
9:  if $\exists$ Projectable Variable $x$ then
10:     if $\exists i s.t. s_i > |R_i|$ then
11:         Return 0,0
12:     end if
13:     Let $Z^0_{\bar{s}} = 1$ if $\bar{s} = \bar{0}$ else 0
14:     Let $S^0_{\bar{s}} = 0 \; \forall \bar{s}$
15:     Let $dom(\bigcup A_{ix}) = \{a_1 \ldots a_n\}$
16:     Let $q = g_1 \ldots g_m$
17:     for $k = 1, n$ do
18:         Let $q' = q[a_k/x]$
19:         $SS = 0$
20:         $ZZ = 0$
21:         Let $R'_i = \{t \in R_i | t.A_{ix} = a_k\}$
22:         for $\bar{j} = 0^m, \bar{s}$ do
23:             Let $\bar{d} = (min\,(|R'_1 \cap G_1|, j_1)\,, \ldots, min\,(|R'_m \cap G_m|, j_m))$
24:             for $\bar{i} = 0^m, \bar{d}$ do
25:                 Let $F = \mathtt{exp}\left(\sum_{p=1,m}(f_p(j_p) - f_p(j_p - i_p))\right)$
26:                 Let $Pk = \Phi_{\bar{R}',\bar{i}}(q')$
27:                 Let $Zk = Z_{\bar{R}',\bar{i}}$
28:                 Let $U = Z^{k-1}_{\bar{j}-\bar{i}} Pk$
29:                 Let $V = S^{k-1}_{\bar{j}-\bar{i}}(Zk - Pk)$
30:                 Let $R = Z^{k-1}_{\bar{j}-\bar{i}} Zk$
31:                 $SS = SS + F(U + V)$
32:                 $ZZ = ZZ + FR$
33:             end for
34:             $S^k_{\bar{j}} = SS$
35:             $Z^k_{\bar{j}} = ZZ$
36:         end for
37:     end for
38: end if
39: Return $S^n_{\bar{s}}$, $Z^n_{\bar{s}}$
```

and $\Phi_{\bar{R},\bar{s}}$ are respectively the sum of potentials of all these worlds. Hence we have

$$\mathtt{Pr}(q) = \frac{\sum_{\bar{s}} \Phi_{\bar{R},\bar{s}}}{\sum_{\bar{s}} Z_{\bar{R},\bar{s}}}$$

Let x be a projectable variable. Then the following proposition establishes the intuition behind projectable variable and homogenization.

PROPOSITION 4.14. *Let A_{ix} be the attribute of R_i where x occurs in g_i. If $R^t_i = \mathtt{hom}(R_i, g_i, \Gamma_i)$, then*

1. *$\mathtt{Pr}(q) = \mathtt{Pr}(q')$ where q' is same as q but over $\bar{R}^t$ instead of $\bar{R}$.*
2. *For any two worlds $W_1 \in \sigma_{A_{ix}=a} R^t_i$ and $W_2 \in \sigma_{A_{ix}=b} R^t_i$, $a \neq b \in dom(x)$*

   $$\Phi(W_1 \wedge W_2) = \Phi(W_1)\Phi(W_2)\mathtt{exp}\left(h_i(|W_1 \cap g_{\Gamma_i}|, |W_2 \cap g_{\Gamma_i}|\right)$$

   where h_i is a real-valued function that depends only on Γ_i.

PROOF. (Sketch) Consider the SCMN corresponding to (R_i, Γ_i). It is composed of independent cliques and R^t_i contains all those cliques which contain tuple that can make g_i true, i.e. they can have an influence on the query q. Since these set of cliques is independent of those in $R_i - R^t_i$, $P(q)$ can be computed just over $\bar{R}^t$. Now consider $\mathcal{M} = SCMN(R^t_i, \Gamma_i)$; By our projectable variable condition either x is a key in which case we say $f_i = 1$. Otherwise no other variable in g_i is a key, which means our homogenization ensures that $\mathcal{M}$ is composed of at most one clique c, with feature function say f_i. So to compute the probability of any world, we just need to keep track of the number of tuples from c. The reader can convince themselves that $h_i(x, y) = f_i(x+y) - f_i(x) - f_i(y)$. $\square$

Algorithm 4.1 first checks for Join or projectable variable condition. The former is trivial. In the latter case, it proceeds by dividing possible worlds of $\bar{R}^t$ into $\sigma_{\bar{A}=a}\bar{R}^t$, where $\bar{A} = \bigcup A_{ix}$ and $a \in \bigcup dom(A_{ix})$. After that the algorithm is similar in spirit to Algorithm 3.1 and due to lack of space we refer the reader to the full version of this paper which mentions the algorithm in detail even for hierarchical constraints.

We should mention that the algorithm can be made more efficient by using some independence properties. For example when $q[a/x]$ and $q[b/x]$ are independent, $\mathtt{Pr}(q[a/x] \vee q[b/x]) = (1 - \mathtt{Pr}(q[a/x]))(1 - \mathtt{Pr}(q[b/x]))$. But it doesn't affect the worst-case complexity and so for the sake of brevity, we haven't written it here.

5. RELATED WORK

Query evaluation over probabilistic databases is a well studied problem. Methods for query evaluation can broadly be classified into two categories: Intensional ([4, 2, 13, 23]) and Extensional ([6, 10, 20, 11]). Our approach belongs to the extensional category.

Intensional methods work by associating with each boolean query a *symbolic event*. Query evaluation is then performed by manipulating expressions over these symbolic events. For example, in [4], lineage is used for defining the symbolic events. In principle, intensional methods can evaluate any given query over a probabilistic database with arbitrary correlations among tuples. However, as the correlations and/or queries become complicated, the symbolic expressions become very large making query evaluation intractable.

On the other hand, extensional methods use efficient operators over real numbers for query evaluation. They work for a restricted set of correlations and queries. Prior work for extensional methods assume very simple correlations like independence ([6, 10]) or exclusions ([3, 20, 11]). As per our knowledge, this is the first paper that uses extensional approach to handle more complicated correlations involving soft constraints.

Query evaluation is closely connected to the inference problem in AI. Many methods proposed in AI literature have been adapted for query evaluation. Deshpande et. al. [24] proposed the use of Markov Networks to represent tuple correlations. In particular, Size-Constrained Markov Networks used in this paper are a subset of correlations they consider. However, they propose an intensional method of query evaluation that makes query evaluation intractable even for safe queries on Size-Constrained Markov Networks. In [15], Gupta et. al. solve an inference problem for Markov Networks that use cardinality based potential functions similar to Size-Constrained Markov Networks. However, they solve the simpler MAP problem that amounts to finding the most likely world among the set of all possible worlds. Query evaluation requires finding the sum of the probabilities of all worlds for a given set of worlds that satisfy a query. Intuitively, the latter is harder because it has to deal with all possible worlds, while the former can make a greedy choice in the selection of its worlds.

For a look at some more recent work on modeling probabilistic databases with graphical models, we refer the reader to [14]. They describe how to represent relational data with Bayesian Networks according to both *possible-worlds* and *domain-frequency* semantics. Our way of representation is different though, as we do not keep a random variable for every attribute of every tuple. Our representation closely resembles that of Markov Logic Networks(MLNs)[22]. An MLN is just a collection of relations and a set of first-order formulas over them with real weights. It gives semantics to these formulas by representing them as features in a markov network over the relations. Our model corresponds to MLNs with formulas like key constraint. But MLNs are a very general model where inference can be very expensive, hence our work also helps to identify some subsets where inference is tractable.

Finally [1] and [27] are some other works which propose handling of constraints during query answering instead of cleaning data apriori. While the former works in a probabilistic setting like us, it enforces *hard* keys. The latter offers a deterministic way of conflict resolution using some form of user specification.

Acknowledgments This work was partially supported by NSF Grants IIS-0513877, IIS-0415193, and IIS-0713576, and by a Gift from Microsoft.

6. REFERENCES

[1] P. Andritsos, A. Fuxman, and R. J. Miller. Clean answers over dirty databases: A probabilistic approach. In *ICDE '06: Proceedings of the 22nd International Conference on Data Engineering*, page 30, Washington, DC, USA, 2006. IEEE Computer Society.

[2] L. Antova, C. Koch, and D. Olteanu. MayBMS: Managing incomplete information with probabilistic world-set decompositions. In *ICDE*, 2007.

[3] D. Barbara, H. Garcia-Molina, and D. Porter. The management of probabilistic data. *IEEE Trans. on Knowledge and Data Eng.*, 1992.

[4] O. Benjelloun, A. D. Sarma, A. Y. Halevy, and J. Widom. ULDBs: Databases with uncertainty and lineage. In *VLDB*, pages 953–964, 2006.

[5] O. Benjelloun, A. D. Sarma, C. Hayworth, and J. Widom. An introduction to ULDBs and the Trio system. *IEEE Data Eng. Bull*, 29(1):5–16, 2006.

[6] R. Cavallo and M. Pittarelli. The theory of probabilistic databases. In *Proceedings of VLDB*, pages 71–81, 1987.

[7] R. Cowell, P. Dawid, S. Lauritzen, and D. Spiegelhalter, editors. *Probabilistic Networks and Expert Systems*. Springer, 1999.

[8] N. Dalvi and D. Suciu. The dichotomy of conjunctive queries on probabilistic structures. In *PODS*, pages 293–302, 2007.

[9] N. Dalvi and D. Suciu. Management of probabilistic data: foundations and challenges. In *PODS '07: Proceedings of the twenty-sixth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 1–12, New York, NY, USA, 2007. ACM Press.

[10] N. N. Dalvi and D. Suciu. Efficient query evaluation on probabilistic databases. *VLDB*, 2004.

[11] N. N. Dalvi and D. Suciu. Management of probabilistic data: foundations and challenges. In *PODS*, 2007.

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

[13] Fuhr, Norbert. A probabilistic relational model for the integration of IR and databases. In *SIGIR*, 1993.

[14] L. Getoor. An introduction to probabilistic graphical models for relational data. *Data Engineering Bulletin*, 29(1), march 2006.

[15] R. Gupta, A. Diwan, and S. Sarawagi. Efficient inference with cardinality-based clique potentials. In *ICML*. ACM, 2007.

[16] I. Ilyas, V. Markl, P. Haas, P. Brown, and A. Aboulnaga. Cords: Automatic discovery of correlations and soft functional dependencies. In *SIGMOD*, pages 647–658, 2004.

[17] J. Pearl. *Probabilistic Reasoning in Intelligent Systems : Networks of Plausible Inference*. Morgan Kaufmann, September 1988.

[18] H. Poon and P. Domingos. Joint inference in information extraction. In *AAAI*, pages 913–918, 2007.

[19] C. Re, N. Dalvi, and D. Suciu. Efficient Top-k query evaluation on probabilistic data. In *ICDE*, 2007.

[20] C. Re, N. N. Dalvi, and D. Suciu. Query evaluation on probabilistic databases. *IEEE Data Eng. Bull*, 2006.

[21] C. Re and D.Suciu. Efficient evaluation of having queries on a probabilistic database. In *Proceedings of DBPL*, 2007.

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

[23] F. Sadri. Integrity constraints in the information source tracking method. *IEEE Transactions on Knowledge and Data Engineering*, 1995.

[24] P. Sen and A. Deshpande. Representing and querying correlated tuples in probabilistic databases. In *ICDE*. IEEE, 2007.

[25] W. Shen, X. Li, and A. Doan. Constraint-based entity matching. In *AAAI*, pages 862–867, 2005.

[26] P. Singla and P. Domingos. Entity resolution with markov logic. In *ICDM*, pages 572–582, 2006.

[27] S. Staworko, J. Chomicki, and J. Marcinkowski. Preference-driven querying of inconsistent relational databases. In *EDBT Workshops*, pages 318–335, 2006.

[28] L. G. Valiant. The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, 8(3):410–421, 1979.