THE NATIONAL UNIVERSITY
of SINGAPORE

School of Computing
Computing 1, 13 Computing Drive, Singapore 117417

TRB3/2014

Conditioning Probabilistic Relational Data with Referential Constraints

Ruiming Tang, Dongxu Shao, M. Lamine Ba and Huayu Wu

March 2014

T e c h n i c a l R e p o r t

Foreword

This technical report contains a research paper, development or tutorial article, which has been submitted for publication in a journal or for consideration by the commissioning organization. The report represents the ideas of its author, and should not be taken as the official views of the School or the University. Any discussion of the content of the report should be sent to the author, at the address shown on the cover.

David ROSENBLUM
Dean of School

Conditioning Probabilistic Relational Data with Referential Constraints

Ruiming Tang[1], Dongxu Shao[1], M. Lamine Ba[2], and Huayu Wu[3]

[1] National University of Singapore, Singapore
{tangruiming,dcsshaod}@nus.edu.sg
[2] Institut Mines–Télécom; Télécom ParisTech; LTCI Paris, France
mouhamadou.ba@telecom-paristech.fr
[3] Institute for Infocomm Research, Singapore
huwu@i2r.a-star.edu.sg

Abstract. A probabilistic relational database is a compact form of a set of deterministic relational databases (namely, *possible worlds*), each of which has a probability. In our framework, the existence of tuples is determined by associated Boolean formulae based on elementary events. An estimation, within such a setting, of the probabilities of possible worlds uses a prior probability distribution specified over the elementary events. Direct observations and general knowledge, in the form of constraints, help refining these probabilities, possibly ruling out some possible worlds. More precisely, new constraints can translate the observation of the existence or non-existence of a tuple, the knowledge of a well-defined rule, such as primary key constraint, foreign key constraint, referential constraint, etc. Informally, the process of enforcing knowledge on a probabilistic database, which consists of computing a new subset of valid possible worlds together with their new (conditional) probabilities, is called *conditioning*. In this paper, we are interested in finding a new probabilistic relational database after conditioning with referential constraints involved. In the most general case, conditioning is intractable. As a result, we restricted our study to probabilistic relational databases in which formulae of tuples are *independent events* in order to achieve some tractability results. We devise and present polynomial algorithms for conditioning probabilistic relational databases with referential constraints.

1 Introduction

Uncertainty of data naturally arises from such applications as information extraction [6], information integration [18, 9] and version control [3], for instance.

Probabilistic databases address the problem of the management and of the representation of uncertain data by means of probabilities. A good probabilistic database model offers a representation of uncertain data that is generally compact and easy to be managed. In a probabilistic database, an instance denotes a set of possible deterministic database instances called *possible worlds*, each of which has a probability that can be initially estimated according to limited prior

knowledge of data. Probabilistic relational databases [4, 16, 10, 8, 15, 2, 5, 12, 11] represent uncertainties at attribute or tuple level, while schema is constrained.

Direct observations and general knowledge, in the form of constraints (such as the existence or non-existence of a tuple, primary key constraint, foreign key constraint, referential constraint, etc), can be enforced into the database during a data cleaning process, for instance. These constraints help refining the probabilities of the possible worlds, possibly ruling out some of them. Enforcing such constraints to the set of possible worlds with their probabilities is called *conditioning* the probabilistic database. The core problem in probabilistic database conditioning is to find a new probabilistic database instance that denotes the subset of possible worlds with their new probabilities corresponding to the conditional probabilities.

The conditioning problem in probabilistic relational databases has been studied in [14] and [17], respectively. Koch and Olteanu [14] show that relational conditioning is NP-Hard. In our previous work [17], we identify tractable scenarios for which we devise polynomial time algorithms. In [17], we focus on the special case in which tuples are independent, and the kinds of constraints considered in the tractable cases studied are observation constraints (i.e. existence constraints) and X-tuple constraints (i.e. primary key constraints).

In this paper, we continue studying the problem of conditioning probabilistic relational data. We adopt the data model in [17] and also focus on the special class of probabilistic relational databases in which the formulae attached to the tuples are *independent events*. We investigate a different kind of constraints, namely referential constraints (also called inclusion dependency [1]), from the ones we studied in [17]. A *referential constraint* is in the form $R[r_1 \dots r_m] \subseteq S[s_1 \dots s_n]$, where R, S are (possibly identical) relation names, $r_1 \dots r_m$ is a subset of distinct attribute names of R and $s_1 \dots s_n$ is a subset of distinct attribute names of S ([1]). This referential constraint is a *foreign key constraint* if $s_1 \dots s_n$ is the primary key of S. A referential constraint in a real-life example is presented in Example 1.

Example 1. There are two relations: Movies (Table 1) and Showings (Table 2). Movies stores a set of movies with movies' title, director and actors. Showings stores locations (theater and hall) where movies are showed. Each tuple is associated with an independent event, representing the probability of this tuple being actual. There is a referential constraint saying that all the showed movies must be produced first, i.e. $Showings[title] \subseteq Movies[title]$. □

The rest of this paper is structured as follows. We start by reviewing state-of-the-art probabilistic relational models and conditioning probabilistic relational databases in Section 2. Then, we present in Section 3 the probabilistic relational data model we consider in this paper. We describe how a referential constraint can be transformed to independent implication constraints and present three classes of implication constraints in Section 4. Tractable algorithms are presented for individual classes of implication constraints in Sections 5, 6 and 7 respectively. The class of constraints in Section 7 covers the first two classes in Sections 5, 6

Title	Director	Actor	event
Hobbit 2	Peter Jackson	Martin Freeman	$e_{(b,1)}$
Hobbit 2	Peter Jackson	Ian McKellen	$e_{(b,2)}$
Hunger Game 2	Francis Lawrence	Jennifer Lawrence	$e_{(b,3)}$
Hunger Game 2	Francis Lawrence	Liam Hemsworth	$e_{(b,4)}$
Fast & Furious 7	James Wan	Vin Diesel	$e_{(b,5)}$
Fast & Furious 7	James Wan	Paul Walker	$e_{(b,6)}$

Table 1. Movies

Theater	Hall	Title	event
Golden Village	hall 1	Hunger Game 2	$e_{(a,1)}$
Golden Village	hall 2	Hunger Game 2	$e_{(a,2)}$
Golden Village	hall 3	Hobbit 2	$e_{(a,3)}$
Golden Village	hall 4	Hobbit 2	$e_{(a,4)}$

Table 2. Showings

and it actually is the class of general referential constraints. We conclude and present future work in Section 8.

2 Related Work

We briefly review hereafter state-of-the-art probabilistic relational data models and conditioning algorithms.

2.1 Probabilistic Relational Models

Probabilistic relational data models can be separated into two groups: tuple-level models and attribute-level models. Granularity of uncertainty differs between the two groups of models.

One simple idea to define a probabilistic relational model is the tuple-mutually-exclusive model ([5]). In this model, a probabilistic database is an ordinary database where each tuple is associated with a probability of being actual. Each tuple is mutually exclusive to any other tuple in the same probabilistic relation. The sum of probabilities of all the tuples in a probabilistic relation is 1. In the tuple-independent model ([7]), a probabilistic database is an ordinary database where each tuple is associated with a probability of being actual, independent from any other tuple. Mixing previous two models is the block-independent-disjoint model ([8, 15, 2]): tuples within a block are mutually-exclusive, while tuples across different blocks are independent.

The expressiveness of block-independent-disjoint model is stronger than tuple-independent model and tuple-mutually-exclusive model. However, it is not expressive enough in some cases, when relationships between tuples, other than independence and mutually exclusion, are needed.

The most expressive model is probabilistic $c-$table (e.g. [12, 13, 11]). In this model, each tuple is associated with a formula constructed from Boolean event

using operators $\wedge, \vee, \neg$. Boolean events are independent and associated with probabilities of being true. A tuple is actual if its associated formula is evaluated to be true.

We adopt the probabilistic relational data model from [17]. It is a tuple-level model with additional constraints integrated as first class citizens, i.e., extended probabilistic $c-$table. The model caters for constraints rather than treating them as add-ons, as other tuple-level models can be adapted to.

2.2 Conditioning

Koch and Olteanu [14] show that relational conditioning is NP-Hard, since the probability computation problem of an arbitrary Boolean expression is hard. They present a general but exponential time algorithm as well as efficient heuristics and decomposition methods.

In [17], we identify tractable scenarios for which we devise polynomial time algorithms. We focus on the special case in which tuples are independent and the tractable cases studied are observation constraints (i.e. existence constraints) and X-tuple constraints (i.e. primary key constraints).

In this paper, we also focus on the special case in which tuples are independent, but we study a different set of constraints, namely referential constraints, from the ones studied in [17].

3 Data model

In [17], we introduced a framework for conditioning probabilistic relational data. In this section, we briefly review the conditioning framework in [17]. In the end of this section, we present two theorems which are not studied in [17].

Let E be a set of symbols called *events (e)*. A *complex event (ce)* is a well formed formula of propositional logic in which events are propositions. $\mathsf{C}(E)$ is the set of complex events formed with the events in E. An *interpretation* of a complex event is a function from E to $\{\mathsf{true}, \mathsf{false}\}$.

Definition 1. *(Probabilistic Relational Database) A* ***probabilistic relational database*** $\mathcal{D}$ *is a quintuple* $\langle D, E, f, C, p\rangle$*: D is a traditional database instance, E is a set of events, f is a function from D to $\mathsf{C}(E)$, C is a subset of $\mathsf{C}(E)$, and p is a function from E to* $[0, 1]$.

The *probability of a complex event* c, noted $p(c)$, is

$$p(c) = \sum_{I \in \mathcal{M}(c)} ((\prod_{I(e)=\mathsf{true}} p(e)) \times (\prod_{I(e)=\mathsf{false}} (1 - p(e))))$$

where $\mathcal{M}(c)$ is the set of models of c.

Informally and in short, a possible world is a traditional relational database such that the complex events associated with the tuples in the possible world are true, the complex events associated with the tuples not in the possible world

are false, and the constraints in C are true. The probability of a possible world is the probability to find such a model given the probabilities of individual events, under the condition that the constraints are held.

Definition 2. *(Possible Worlds) Let $\mathcal{D} = \langle D, \mathsf{E}, f, C, p\rangle$ be a probabilistic relational database. D' is a **possible world** of $\mathcal{D}$ if and only if there exists a model of the following formula F with a non zero probability.*

$$F = (\bigwedge_{t_i \in D'} f(t_i)) \wedge (\bigwedge_{t_i \notin D'} \neg f(t_i)) \wedge (\bigwedge_{c \in C} c) \quad \textit{and} \quad p(F) \neq 0$$

We call $p_{\mathcal{D}}(D')$ the probability, $p(F|C)$, of the possible world D' in the probabilistic relational database $\mathcal{D}$. We call $\mathfrak{P}(\mathcal{D})$ the set of possible worlds of $\mathcal{D}$.

$\mathcal{D}$ is **consistent** (resp. **inconsistent**) if and only if there exists at least one possible world (resp. there does not exist a possible world) of $\mathcal{D}$, i.e. $\mathfrak{P}(\mathcal{D}) \neq \emptyset$ (resp. $\mathfrak{P}(\mathcal{D}) = \emptyset$). $\mathcal{D}$ is inconsistent iff C is always evaluated to be false. Note that we define the concept of "consistent" only for defining the conditioning problem.

We defined an equivalent relationship on probabilistic relational databases. Two probabilistic relational databases are *world-equivalent* if they have the same possible worlds with the same probabilities, i.e. $\mathcal{D}_1 \equiv_w \mathcal{D}_2$ if and only if:

$$\forall D' \subseteq D \; ((D' \in \mathfrak{P}(\mathcal{D}_1)) \leftrightarrow (D' \in \mathfrak{P}(\mathcal{D}_2))) \qquad \textit{and}$$

$$\forall D' \in \mathfrak{P}(\mathcal{D}_1)(p_{\mathcal{D}_1}(D') = p_{\mathcal{D}_2}(D'))$$

We showed in [17] that for any consistent probabilistic relational database $\mathcal{D}_1$ then there exists a probabilistic relational database $\mathcal{D}_2 = \langle D, E_2, f_2, \emptyset, p_2\rangle$ such that $\mathcal{D}_2 \equiv_w \mathcal{D}_1$.

Conditioning a probabilistic relational database consists in adding constraints. The **conditioning problem** consists in finding a world-equivalent probabilistic relational database with no constraint, given one probabilistic relational database with constraints.

We prove below two propositions which are not studied in [17].

Proposition 1. *Let $\mathcal{D}_1 = \langle D, E_1, f_1, C, p_1\rangle$ be a consistent probabilistic relational database. Let $\mathcal{D}_2 = \langle D, E_2, f_2, \emptyset, p_2\rangle$ be a probabilistic relational database such that $\mathcal{D}_1 \equiv_w \mathcal{D}_2$. For a tuple $t \in D$, if $f_1(t)$ is independent of C, it is possible to condition so that $p_1(f_1(t)) = p_2(f_2(t))$ and it is possible to have $f_2(t) = f_1(t)$.*

Proof.

$$p_2(f_2(t)) = p_1(f_1(t)|C) = \frac{p_1(f_1(t) \wedge C)}{p_1(C)} = \frac{p_1(f_1(t)) \cdot p_1(C)}{p_1(C)} = p_1(f_1(t))$$

Since the formula of t is independent of C, it is possible to keep it unchanged.

For the other tuples in D (whose associated formulae are dependent on C), their formula have to be updated and the probabilities of new created events have to be defined. □

In this paper, we consider a special class of probabilistic relational databases, namely *probabilistic relational databases with independent events* as the input of the conditioning problem.

Proposition 2. *Let $\mathcal{D} =< D, E_1, f_1, C, p_1 >$ be a consistent probabilistic relational database with independent events. For a tuple $t \in D$, if $f_1(t)$ is independent of C, it is possible to condition so that its formula $f_2(t)$ is also an unique independent event and $p_1(f_1(t)) = p_2(f_2(t))$.*

Proposition 2 is a direct consequence of Proposition 1.

4 Referential Constraints

In this section, we first present how a referential constraint can be transformed to a set of independent implication constraints and we introduce the three kinds of implication constraints for which we will then present algorithms in the following sections. We then introduce one particular technical tool which relates to the *relevant part* of a probabilistic relational database for a given constraint. We also present the notion of *possible worlds according to the relevant part* of a probabilistic relational database for a constraint.

4.1 Transforming Referential Constraints to Implication Constraints

We start this section by defining the implication constraints.

Definition 3. *(Implication Constraints) An **implication constraint** is a well formed formula of propositional logic whose format is*

$$\bigvee_{i=1}^{m} e_{(a,i)} \Rightarrow \bigvee_{i=1}^{n} e_{(b,i)}$$

A referential constraint can be transformed to a set of independent implication constraints. We illustrate this remark by using the example below.

Example 2. Consider the referential constraint in Example 1, i.e., all the showed movies must be produced first, i.e. $Showings[title] \subseteq Movies[title]$. In this example, the referential constraint is transformed to two independent implication constraints. If "Hobbit 2" is showed, then it must be produced, i.e. $(e_{(a,3)} \vee e_{(a,4)}) \Rightarrow (e_{(b,1)} \vee e_{(b,2)})$. If "Hunger Game 2" is showed, then it must be produced, i.e. $(e_{(a,1)} \vee e_{(a,2)}) \Rightarrow (e_{(b,3)} \vee e_{(b,4)})$. Note that this result in only under the probabilistic relational model with independent events. □

In the next section, we will prove that conditioning two independent constraints yields the same result as conditioning them separately (Theorem 2), therefore **we consider conditioning a single implication constraint in the rest of the paper**. Note that when there are multiple referential constraints or

there are multiple *dependent* implication constraints, our algorithms (which will be presented later) cannot be applied to solve the conditioning problem, because tuples affected by the constraints may have complicated formulae to make our algorithm fail to work.

Given a referential constraint $R[r_1, r_2, ..., r_m] \subseteq S[s_1, s_2, ..., s_n]$, we present below three classes of implication constraints (in the form of $\bigvee_{i=1}^{m} f_1(t_{(a,i)}) \Rightarrow \bigvee_{i=1}^{n} f_1(t_{(b,i)})$). In the rest of this paper, we use A to represent the set of tuples $\{t_{(a,1)}, ..., t_{(a,m)}\}$ and we use B to present the set of tuples $\{t_{(b,1)}, ..., t_{(b,n)}\}$.

- when $r_1, r_2, ..., r_m$ is the primary key of R and $s_1, s_2, ..., s_n$ is the primary key of S, we have $A = \{t_{(a,1)}\}, B = \{t_{(b,1)}\}$. We refer to this class as FKPK constraints. This class of constraints is realistic when relation R is a subclass of relation S.
- when $s_1, s_2, ..., s_n$ is the primary key of S, we have $A = \{t_{(a,1)}, ..., t_{(a,m)}\}, B = \{t_{(b,1)}\}$. This is the class of foreign key constraints, and we refer to this class as FK constraints.
- for a general referential constraint, we have $A = \{t_{(a,1)}, ..., t_{(a,m)}\}, B = \{t_{(b,1)}, ..., t_{(b,n)}\}$. This is the **most general class of referential constraints** and we refer to this class as REF constraints.

4.2 Local Database and Local Possible Worlds

Corollary 1. *Let $\mathcal{D} = \langle D, E_1, f_1, C, p_1\rangle$ be a consistent probabilistic relational database with independent events and C be an implication constraint. After conditioning, (1) the formulae of all the tuples in A and B must be updated and the probabilities of new events must be defined; (2) for the other tuples, the formulae and probabilities are unmodified.*

This corollary can be deduced from Proposition 2, because if $t \notin A \cup B$, then $f_1(t)$ is independent of C.

Definition 4. *(Local Database) Let $\mathcal{D} = \langle D, E_1, f_1, C, p_1\rangle$ be a consistent probabilistic relational database with independent events and C be an implication constraint over two sets of tuples[4] A, B. The **local database**, that we denote by $LD(C, D)$ of D with respect to C is $A \cup B$.*

Theorem 1. *Let $\mathcal{D}_1 = \langle D, E_1, f_1, C, p_1\rangle$ be a consistent probabilistic relational database with independent events and $\mathcal{D}_2 = \langle D, E_2, f_2, \emptyset, p_2\rangle$ be a probabilistic relational database. C is an implication constraint. We claim that $\mathcal{D}_1 \equiv_w \mathcal{D}_2$ iff $\langle LD(C, D), E_1, f_1, C, p_1\rangle \equiv_w \langle LD(C, D), E_2, f_2, \emptyset, p_2\rangle$.*

Theorem 2. *Let $\mathcal{D}_1 = \langle D, E_1, f_1, \{C_1, C_2\}, p_1\rangle$ be a consistent probabilistic relational database with independent events, where C_1, C_2 are two independent implication constraints. $\mathcal{D}_2 = \langle D, E_2, f_2, \emptyset, p_2\rangle$ is a probabilistic relational database. We claim that $\mathcal{D}_2 \equiv_w \mathcal{D}_1$ iff $\langle LD(C_1, D), E_1, f_1, C_1, p_1\rangle \equiv_w \langle LD(C_1, D), E_2, f_2, \emptyset, p_2\rangle$ and $\langle LD(C_2, D), E_1, f_1, C_2, p_1\rangle \equiv_w \langle LD(C_2, D), E_2, f_2, \emptyset, p_2\rangle$.*

[4] We do not distinguish tuples and their associated events because we study the probabilistic relational model with independent events.

Proof. Since C_1, C_2 are independent, and the events of tuples are also independent, we deduce that $\mathsf{LD}(C_1, D) \cap \mathsf{LD}(C_2, D) = \emptyset$. Moreover, due to Proposition 2, the formulae of tuples in $\mathsf{LD}(C_1, D)$ are not updated by enforcing C_2, and conversely the formulae of tuples in $\mathsf{LD}(C_2, D)$ are not changed by enforcing C_1. Therefore we can enforce C_1, C_2 separately without affecting each other. Based on Theorem 1, we can prove the desired claim. □

Definition 5. *(Local Possible Worlds) The* ***local possible worlds*** *of* $\mathcal{D}_1 = \langle D, E_1, f_1, C, p_1 \rangle$ *correspond to the possible worlds of* $\langle \mathsf{LD}(C, D), E_1, f_1, C, p_1 \rangle$.

If the number of local possible worlds is polynomial to the size of the local database, a P-TIME conditioning algorithm is trivial to devise by enumerating all the local possible worlds. While the number of local possible worlds is exponential to the size of the local database, finding a P-TIME conditioning algorithm is a challenge. Therefore, before going to study conditioning algorithms, we show the numbers of local possible worlds for different classes of implication constraints.

For an FKPK constraint, the number of local possible worlds is 3: (1) 1 local possible worlds when $t_{(a,1)}, t_{(b,1)}$ exist; (2) 1 local possible worlds when $t_{(a,1)}$ does not exist and $t_{(b,1)}$ exists; (3) 1 local possible worlds when neither of $t_{(a,1)}, t_{(b,1)}$ exist.

For an FK constraint, the number of local possible worlds is $2^m + 1$, which is exponential to the number of tuples in the local database: (1) when at least one of $\{t_{(a,1)}, ..., t_{(a,m)}\}$ exists and $t_{(b,1)}$ exists, there are $2^m - 1$ local possible worlds; (2) when $\{t_{(a,1)}, ..., t_{(a,m)}\}$ does not exist and $t_{(b,1)}$ exists, there is 1 local possible worlds; (3) when $\{t_{(a,1)}, ..., t_{(a,m)}\}$ does not exist and $t_{(b,1)}$ does not exist, there is 1 local possible worlds.

For a REF constraint, the number of local possible worlds is $2^n + (2^m - 1)(2^n - 1)$, which is exponential to the number of tuples in the local database: (1) when at least one of $\{t_{(a,1)}, ..., t_{(a,m)}\}$ exists and at least one of $\{t_{(b,1)}, ..., t_{(b,n)}\}$ exists, there are $(2^m - 1)(2^n - 1)$ local possible worlds; (2) when $\{t_{(a,1)}, ..., t_{(a,m)}\}$ does not exist and at least one of $\{t_{(b,1)}, ..., t_{(b,n)}\}$ exists, there are $2^n - 1$ local possible worlds; (3) when $\{t_{(a,1)}, ..., t_{(a,m)}\}$ does not exist and $\{t_{(b,1)}, ..., t_{(b,n)}\}$ does not exist as well, there is 1 local possible world.

5 Conditioning Algorithm for FKPK Constraints

In this section, we present a conditioning algorithm for FKPK constraints. In an FKPK constraint, $A = \{t_i\}$ and $B = \{t_j\}$. Assume $f_1(t_i) = e_i$ and $f_1(t_j) = e_j$. The local database is $\mathsf{LD}(C, D) = \{t_i, t_j\}$.

The implication constraint is formulated as $e_i \Rightarrow e_j$. Its probability is computed as

$$p_1(C) = p_1(\neg e_i \vee e_j) = p_1(e_j) + (1 - p_1(e_i))(1 - p_1(e_j))$$

The probability of such a constraint can be computed in linear time. Algorithm 1 presents the conditioning algorithm for FKPK constraints. It introduces

Algorithm 1: Conditioning algorithm for FKPK implication constraints

Data: $\langle \mathsf{LD}(C, D), E_1, f_1, C, p_1 \rangle$
Result: A world equivalent $\langle \mathsf{LD}(C, D), E_2, f_2, \emptyset, p_2 \rangle$

1 $f_2(t_i) \leftarrow x_i$, $f_2(t_j) \leftarrow x_i \vee x_j$;
2 $p_2(x_i) = \frac{p_1(e_i)p_1(e_j)}{p_1(C)}$, $p_2(x_j) = p_1(e_j)$;

2 events for the tuples. The time complexity of Algorithm 1 is linear to the size of the local database.

Theorem 3. *Algorithm 1 is correct and performs in linear time.*

Proof. There are three possible worlds of $\langle \mathsf{LD}(C, D), E_1, f_1, C, p_1 \rangle$: in K_1, neither of t_i and t_j exists; in K_2, only t_j exists; in K_3, both of t_i and t_j exist. After conditioning, $\langle \mathsf{LD}(C, D), E_2, f_2, \emptyset, p_2 \rangle$ has the same three possible worlds. We are going to prove that for each possible world, its probability after conditioning is the same as before conditioning.

- For K_1,

$$p_2(K_1) = p_2(\neg x_i \wedge \neg x_j) = (1 - p_2(x_i))(1 - p_2(x_j))$$
$$= \frac{p_1(C) - p_1(e_i)p_1(e_j)}{p_1(C)}(1 - p_1(e_j)) = \frac{(1 - p_1(e_i))(1 - p_1(e_j))}{p_1(C)} = p_1(K_1)$$

- For K_2,

$$p_2(K_2) = p_2(\neg x_i \wedge x_j) = (1 - p_2(x_i))p_2(x_j)$$
$$= \frac{p_1(C) - p_1(e_i)p_1(e_j)}{p_1(C)}p_1(e_j) = \frac{(1 - p_1(e_i))p_1(e_j)}{p_1(C)} = p_1(K_2)$$

- For K_3,

$$p_2(K_3) = p_2(x_i) = \frac{p_1(e_i)p_1(e_j)}{p_1(C)} = p_1(K_3)$$

Algorithm 1 performs in linear time since it introduces two events and their probabilities can be computed in constant time. □

6 Conditioning Algorithm for FK Constraints

In this section, we present a conditioning algorithm for FK constraints. In an FK constraint, $A = \{t_{(a,1)}, ..., t_{(a,m)}\}$ and $B = \{t_{(b,1)}\}$. Assume $f_1(t_i) = e_i$. The local database is $\mathsf{LD}(C, D) = \{t_{(a,1)}, ..., t_{(a,m)}, t_{(b,1)}\}$.

Algorithm 2: Conditioning algorithm for FK implication constraints

Data: $\langle \mathsf{LD}(C, D), E_1, f_1, C, p_1 \rangle$
Result: A world equivalent $\langle \mathsf{LD}(C, D), E_2, f_2, \emptyset, p_2 \rangle$

foreach $i \in [1, m]$ **do**
  $f_2(t_{(a,i)}) \leftarrow \eta \wedge x_i$;
$f_2(t_{(b,1)}) \leftarrow (\eta \wedge \bigvee_{i=1}^{m} x_i) \vee \lambda$
foreach $i \in [1, m]$ **do**
  $p_2(x_i) = p_1(e_{(a,i)})$;
$p_2(\lambda) = p_1(e_{(b,1)}), p_2(\eta) = \frac{p_1(e_{(b,1)})}{p_1(C)}$;

The constraint is formulated as $\bigvee_{i=1}^{m} e_{(a,i)} \Rightarrow e_{(b,1)}$. Its probability is computed as

$$
\begin{aligned}
p_1(C) &= p_1(\neg \bigvee_{i=1}^{m} e_{(a,i)} \vee e_{(b,1)}) = p_1(\bigwedge_{i=1}^{m} \neg e_{(a,i)} \vee e_{(b,1)}) \\
&= p_1(\bigwedge_{i=1}^{m} \neg e_{(a,i)}) + p_1(e_{(b,1)}) - p_1(\bigwedge_{i=1}^{m} \neg e_{(a,i)} \wedge e_{(b,1)}) \\
&= \prod_{i=1}^{m} (1 - p_1(e_{(a,i)})) + p_1(e_{(b,1)}) - \prod_{i=1}^{m} (1 - p_1(e_{(a,i)})) p_1(e_{(b,1)}) \\
&= p_1(e_{(b,1)}) + (1 - p_1(e_{(b,1)})) \prod_{i=1}^{m} (1 - p_1(e_{(a,i)}))
\end{aligned}
$$

The probability of such a constraint can be computed in linear time. Algorithm 2 presents the conditioning algorithm for FK constraints. It introduces $m + 2$ events. The time complexity of Algorithm 2 is linear to the size of the local database.

Theorem 4. *Algorithm 2 is correct and performs in linear time to size of the local database.*

Proof. There are $2^m + 1$ possible worlds of $\langle \mathsf{LD}(C, D), E_1, f_1, C, p_1 \rangle$: in K_1, there is no tuple; in K_2, there is only $t_{(b,1)}$ exists; in the other $2^m - 1$ possible worlds, $t_{(b,1)}$ exists and at least one of $t_{(a,1)}, ..., t_{(a,m)}$ exists. After conditioning, $\langle \mathsf{LD}(C, D), E_2, f_2, \emptyset, p_2 \rangle$ has the same possible worlds. We are going to prove that for each possible world, its probability after conditioning is the same as before conditioning.

– For K_1,

$$p_2(K_1) = p_2(\neg\lambda \wedge (\neg\eta \vee \eta \wedge \bigwedge_{i=1}^{m} \neg x_i)) = (1 - p_2(\lambda))(1 - p_2(\eta) + p_2(\eta) \prod_{i=1}^{m}(1 - p_2(x_i)))$$

$$= (1 - p_1(e_{(b,1)}))(1 - \frac{p_1(e_{(b,1)})}{p_1(C)} + \frac{p_1(e_{(b,1)})}{p_1(C)} \prod_{i=1}^{m}(1 - p_1(e_{(a,i)})))$$

$$= \frac{1 - p_1(e_{(b,1)})}{p_1(C)}(p_1(C) - p_1(e_{(b,1)}) + p_1(e_{(b,1)}) \prod_{i=1}^{m}(1 - p_1(e_{(a,i)})))$$

$$= \frac{1 - p_1(e_{(b,1)})}{p_1(C)} \prod_{i=1}^{m}(1 - p_1(e_{(a,i)})) = p_1(K_1)$$

– For K_2, the proof is similar to K_1, except that λ is true in K_2.
– For the other $2^m - 1$ possible worlds, their proofs are similar, therefore we only show one of them. Consider the possible world K that $t_{(b,1)}$ exists, $t_{(a,k)}$ exists and all the others do not.

$$p_2(K) = p_2(\eta \wedge x_k \wedge \bigwedge_{i=1, i\neq k}^{m} \neg x_i) = p_2(\eta)p_2(x_k) \prod_{i=1, i\neq k}^{m} (1 - p_2(x_i))$$

$$= \frac{p_1(e_{(b,1)})}{p_1(C)} p_1(e_{(a,k)}) \prod_{i=1, i\neq k}^{m} (1 - p_1(e_{(a,i)})) = p_1(K)$$

Algorithm 2 performs in linear time since it introduces $m + 2$ events and their probabilities can be computed in constant time. □

We illustrate Algorithm 2 by an example.

Example 3. $A = \{t_{(a,1)}, t_{(a,2)}, t_{(a,3)}\}$ and $B = \{t_{(b,1)}\}$. $f_1(t_{(a,1)}) = e_{(a,1)}, f_1(t_{(a,2)}) = e_{(a,2)}, f_1(t_{(a,3)}) = e_{(a,3)}, f_1(t_{(b,1)}) = e_{(b,1)}$. The FK implication constraint is $(e_{(a,1)} \vee e_{a,2} \vee e_{a,3}) \Rightarrow e_{(b,1)}$.

After conditioning, $f_2(t_{(a,1)}) = \eta \wedge x_1, f_2(t_{(a,2)}) = \eta \wedge x_2, f_2(t_{(a,3)}) = \eta \wedge x_3, f_2(t_{(b,1)}) = (\eta \wedge (x_1 \vee x_2 \vee x_3)) \vee \lambda$. The probabilities of new events are $p_2(x_1) = p_1(e_{(a,1)}), p_2(x_2) = p_1(e_{(a,2)}), p_2(x_3) = p_1(e_{(a,3)}), p_2(\lambda) = p_1(e_{(b,1)}), p_2(\eta) = \frac{p_1(e_{(b,1)})}{p_1(C)}$. □

7 Conditioning Algorithm for REF Constraints

In this section, we present conditioning algorithms for REF constraints.

7.1 Simplified Case

To understand the conditioning algorithm for REF constraints better, we consider a simplified version of REF constraint first. In this case, $A = \{t_{(a,1)}\}$ and

Algorithm 3: Conditioning algorithm for simplified REF implication constraints

Data: $\langle \mathsf{LD}(C, D), E_1, f_1, C, p_1 \rangle$
Result: A world equivalent $\langle \mathsf{LD}(C, D), E_2, f_2, \emptyset, p_2 \rangle$

1 $f_2(t_{(a,1)}) \leftarrow \lambda$
2 $f_2(t_{(b,1)}) \leftarrow (\lambda \wedge x_1) \vee y_1$
3 **foreach** $i \in [2, n-1]$ **do**
4 $\quad f_2(t_{(b,i)}) \leftarrow (\lambda \wedge \bigwedge_{j=1}^{i-1}(\neg x_j \wedge \neg y_j) \wedge x_i) \vee y_i$;
5 $f_2(t_{(b,n)}) \leftarrow (\lambda \wedge \bigwedge_{j=1}^{n-1}(\neg x_j \wedge \neg y_j)) \vee y_n$;
6 **foreach** $i \in [1, n]$ **do**
7 $\quad p_2(y_i) = p_1(e_{(b,i)})$;
8 **foreach** $i \in [1, n-1]$ **do**
9 $\quad p_2(x_i \vee y_i) = \frac{p_1(e_{(b,i)})}{p_1(\bigvee_{j=i}^{n} e_{(b,j)})}$;
10 $p_2(\lambda) = \frac{p_1(e_{(a,1)}) p_1(\bigvee_{i=1}^{n} e_{(b,i)})}{p_1(C)}$;

$B = \{t_{(b,1)}, ..., t_{(b,n)}\}$. Assume $f_1(t_i) = e_i$. The local database is $\mathsf{LD}(C, D) = \{t_{(a,1)}, t_{(b,1)}, ..., t_{(b,n)}\}$.

The constraint is formulated as $e_{(a,1)} \Rightarrow \bigvee_{i=1}^{n} e_{(b,i)}$. Its probability is computed as

$$p_1(C) = p_1(\neg e_{(a,1)} \vee \bigvee_{i=1}^{n} e_{(b,i)}) = 1 - p_1(e_{(a,1)}) + p_1(e_{(a,1)}) p(\bigvee_{i=1}^{n} e_{(b,i)})$$

The probability of such a constraint can be computed in linear time, since the probability of a disjunction of independent events can be computed associatively. Algorithm 3 presents the conditioning algorithm for simplified REF constraints. It introduces $2n$ events. The time complexity of Algorithm 3 is linear to the size of the local database.

Theorem 5. *Algorithm 3 is correct and performs in linear time to size of the local database.*

We omit the proof since it is similar to the proof of previous theorems. The basic idea of the proof is to prove for every possible world, its probability is the same after conditioning as before. Moreover, we have to make sure all the probability values are valid, i.e,

- $p_2(x_i \vee y_i) \in [0, 1]$.
  It is true, because $p_1(e_{(b,i)}) \leq p_1(\bigvee_{j=i}^{n} e_{(b,j)})$ and from line 9 of Algorithm 3 we know that $p_2(x_i \vee y_i) \in [0, 1]$.
- $p_2(x_i \vee y_i) \geq p_2(y_i)$.
  It is true, because $p_1(\bigvee_{j=i}^{n} e_{(b,j)}) \leq 1$, and from line 7, 9 of Algorithm 3 we know that $p_2(x_i \vee y_i) \geq p_2(y_i)$.

– $p_2(\lambda) \in [0,1]$.
  It is true, because $p_1(e_{(a,1)})p_1(\bigvee_{i=1}^{n} e_{(b,i)}) \leq p_1(C)$ (from the formulae of C) and from line 10 of Algorithm 3 we know that $p_2(\lambda) \in [0,1]$.

We illustrate Algorithm 3 by the example below.

Example 4. $A = \{t_{(a,1)}\}$ and $B = \{t_{(b,1)}, t_{(b,2)}, t_{(b,3)}\}$. $f_1(t_{(a,1)}) = e_{(a,1)}, f_1(t_{(b,1)}) = e_{(b,1)}, f_1(t_{(b,2)}) = e_{(b,2)}, f_1(t_{(b,3)}) = e_{(b,3)}$. The simplified REF implication constraint is $e_{(a,1)} \Rightarrow (e_{(b,1)} \vee e_{b,2} \vee e_{b,3})$.

After conditioning, $f_2(t_{(a,1)}) = \lambda, f_2(t_{(b,1)}) = (\lambda \wedge x_1) \vee y_1, f_2(t_{(b,2)}) = (\lambda \wedge \neg x_1 \wedge \neg y_1 \wedge x_2) \vee y_2, f_2(t_{(b,3)}) = (\lambda \wedge \neg x_1 \wedge \neg y_1 \wedge \neg x_2 \wedge \neg y_2) \vee y_3$. The probabilities of new events are $p_2(y_1) = p_1(e_{(b,1)}), p_2(y_2) = p_1(e_{(b,2)}), p_2(y_3) = p_1(e_{(b,3)}), p_2(x_1 \vee y_1) = \frac{p_1(e_{(b,1)})}{p_1(e_{(b,1)} \vee e_{(b,2)} \vee e_{(b,3)})}, p_2(x_2 \vee y_2) = \frac{p_1(e_{(b,2)})}{p_1(e_{(b,2)} \vee e_{(b,3)})}, p_2(\lambda) = \frac{p_1(e_{(a,1)})p_1(e_{(b,1)} \vee e_{(b,2)} \vee e_{(b,3)})}{p_1(C)}$. □

7.2 General Case

In a REF constraint, $A = \{t_{(a,1)}, ..., t_{(a,m)}\}$ and $B = \{t_{(b,1)}, ..., t_{(b,n)}\}$. Assume $f_1(t_i) = e_i$. The local database is $\mathsf{LD}(C, D) = \{t_{(a,1)}, ..., t_{(a,m)}, t_{(b,1)}, ..., t_{(b,n)}\}$.

The constraint is formulated as $\bigvee_{i=1}^{m} e_{(a,i)} \Rightarrow \bigvee_{i=1}^{n} e_{(b,i)}$. Its probability is computed as

$$p_1(C) = p_1(\neg \bigvee_{i=1}^{m} e_{(a,i)} \vee \bigvee_{i=1}^{n} e_{(b,i)}) = p_1(\bigwedge_{i=1}^{m} \neg e_{(a,i)} \vee \bigvee_{i=1}^{n} e_{(b,i)})$$
$$= \prod_{i=1}^{m}(1 - p_1(e_{(a,i)})) + p_1(\bigvee_{i=1}^{n} e_{(b,i)})p_1(\bigvee_{i=1}^{m} e_{(a,i)})$$

The probability of such a constraint can be computed in linear time, since the probability of a disjunction of independent events can be computed associatively. Algorithm 4 presents the conditioning algorithm for simplified REF constraints. It introduces $m + 2n$ events. The time complexity of Algorithm 4 is linear to the size of the local database.

Theorem 6. *Algorithm 4 is correct and performs in linear time to size of the local database.*

We omit the proof since it is similar to the proof of previous theorems. The basic idea of the proof is to prove for every possible world, its probability is the same after conditioning as before. Moreover, we have to make sure all the probability values are valid, i.e,

– $p_2(x_i \vee y_i) \in [0,1]$.
  It is true for the same reason as the previous section.
– $p_2(x_i \vee y_i) \geq p_2(y_i)$.
  It is true for the same reason as the previous section.

Algorithm 4: Conditioning algorithm for REF implication constraints

Data: $\langle \mathsf{LD}(C, D), E_1, f_1, C, p_1 \rangle$
Result: A world equivalent $\langle \mathsf{LD}(C, D), E_2, f_2, \emptyset, p_2 \rangle$

foreach $i \in [1, m]$ **do**
  $f_2(t_{(a,i)}) \leftarrow \lambda \wedge a_i$
$f_2(t_{(b,1)}) \leftarrow (\lambda \wedge \bigvee_{j=1}^{m} a_j \wedge x_1) \vee y_1$
foreach $i \in [2, n-1]$ **do**
  $f_2(t_{(b,i)}) \leftarrow (\lambda \wedge \bigvee_{j=1}^{m} a_j \wedge \bigwedge_{j=1}^{i-1}(\neg x_j \wedge \neg y_j) \wedge x_i) \vee y_i$;
$f_2(t_{(b,n)}) \leftarrow (\lambda \wedge \bigvee_{j=1}^{m} a_j \wedge \bigwedge_{j=1}^{n-1}(\neg x_j \wedge \neg y_j)) \vee y_n$;
foreach $i \in [1, m]$ **do**
  $p_2(a_i) = p_1(e_{(a,i)})$;
foreach $i \in [1, n]$ **do**
  $p_2(y_i) = p_1(e_{(b,i)})$;
foreach $i \in [1, n-1]$ **do**
  $p_2(x_i \vee y_i) = \frac{p_1(e_{(b,i)})}{p_1(\bigvee_{j=i}^{n} e_{(b,j)})}$;
$p_2(\lambda) = \frac{p_1(\bigvee_{i=1}^{n} e_{(b,i)})}{p_1(C)}$;

– $p_2(\lambda) \in [0, 1]$.
  It is true, because we can deduce $p_1(\bigvee_{i=1}^{n} e_{(b,i)}) \leq p_1(C)$ from

$$p_1(C) = p_1(\bigvee_{i=1}^{n} e_{(b,i)}) + \prod_{i=1}^{m}(1 - p_1(e_{(a,i)})) \prod_{i=1}^{n}(1 - p_1(e_{(b,i)}))$$

We illustrate Algorithm 4 using the example below.

Example 5. $A = \{t_{(a,1)}, t_{(a,2)}\}$ and $B = \{t_{(b,1)}, t_{(b,2)}\}$. $f_1(t_{(a,1)}) = e_{(a,1)}, f_1(t_{(a,2)}) = e_{(a,2)}, f_1(t_{(b,1)}) = e_{(b,1)}, f_1(t_{(b,2)}) = e_{(b,2)}$. The REF implication constraint is $(e_{(a,1)} \vee e_{(a,2)}) \Rightarrow (e_{(b,1)} \vee e_{b,2})$.

After conditioning, $f_2(t_{(a,1)}) = \lambda \wedge a_1, f_2(t_{(a,2)}) = \lambda \wedge a_2, f_2(t_{(b,1)}) = (\lambda \wedge (a_1 \vee a_2) \wedge x_1) \vee y_1, f_2(t_{(b,2)}) = (\lambda \wedge (a_1 \vee a_2) \wedge \neg x_1 \wedge \neg y_1) \vee y_2$. The probabilities of new events are $p_2(y_1) = p_1(e_{(b,1)}), p_2(y_2) = p_1(e_{(b,2)}), p_2(a_1) = p_1(e_{(a,1)}), p_2(a_2) = p_1(e_{(a,2)}), p_2(x_1 \vee y_1) = \frac{p_1(e_{(b,1)})}{p_1(e_{(b,1)} \vee e_{(b,2)})}, p_2(\lambda) = \frac{p_1(e_{(b,1)} \vee e_{(b,2)})}{p_1(C)}$. □

8 Conclusion and Future Work

In this paper, we have studied the problem of conditioning probabilistic relational data with referential constraints. We focus on the special case of probabilistic relational databases with independent events. We present and devise tractable algorithms for three classes of implication constraints, namely FKPK, FK and REF. REF is the class of general referential constraints and covers FKPK and FK constraints.

A probabilistic relational model captures only the uncertainty of data value, while probabilistic XML, as a hierarchical data model, captures the uncertainty

of both value and structure. The uncertainty of the structure of data introduces new challenges for the conditioning problem. The implication constraints are also practical in probabilistic XML data, i.e., existence of a set of nodes implies existence of another set of nodes. We are now studying conditioning probabilistic XML data with implication constraints.

Acknowledgment. This paper is partially supported by the French government under the STIC-Asia program, CCIPX project. The work by Huayu Wu is supported by the A*STAR SERC Grant No. 1224200004.

References

1. S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.
2. P. Agrawal, O. Benjelloun, A. D. Sarma, C. Hayworth, S. Nabar, T. Sugihara, and J. Widom. Trio: A system for data, uncertainty, and lineage. In *VLDB*, 2006.
3. M. L. Ba, T. Abdessalem, and P. Senellart. Uncertain version control in open collaborative editing of tree-structured documents. In *Proc. DocEng*, Florence, Italy, 2013.
4. D. Barbará, H. Garcia-Molina, and D. Porter. The management of probabilistic data. *Proc. IEEE Trans. Knowl. Data Eng.*, 1992.
5. R. Cavallo and M. Pittarelli. The theory of probabilistic databases. In *Proc. VLDB*, 1987.
6. C.-H. Chang, M. Kayed, M. R. Girgis, and K. F. Shaalan. A survey of Web information extraction systems. *IEEE Trans. on Knowl. and Data Eng.*, 2006.
7. N. N. Dalvi and D. Suciu. Efficient query evaluation on probabilistic databases. In *VLDB*, pages 864–875, 2004.
8. D. Dey and S. Sarkar. Psql: A query language for probabilistic relational data. *Data Knowl. Eng.*, 28(1):107–120, 1998.
9. X. L. Dong, A. Halevy, and C. Yu. Data integration with uncertainty. *VLDB Journal*, 2009.
10. T. Eiter, T. Lukasiewicz, and M. Walter. A data model and algebra for probabilistic complex values. *Proc. Ann. Math. Artif. Intell.*, 2001.
11. R. Fink, D. Olteanu, and S. Rath. Providing support for full relational algebra in probabilistic databases. In *ICDE*, pages 315–326, 2011.
12. N. Fuhr and T. Rölleke. A probabilistic relational algebra for the integration of information retrieval and database systems. *ACM Trans. Inf. Syst.*, 15(1), 1997.
13. T. J. Green and V. Tannen. Models for incomplete and probabilistic information. In *EDBT Workshops*, pages 278–296, 2006.
14. C. Koch and D. Olteanu. Conditioning probabilistic databases. *Proc. PVLDB*, 2008.
15. C. Re and D. Suciu. Materialized views in probabilistic databases for information exchange and query optimization. In *VLDB*, pages 51–62, 2007.
16. D. Suciu, D. Olteanu, C. Ré, and C. Koch. *Probabilistic Databases*. Morgan & Claypool, 2011.
17. R. Tang, R. Cheng, H. Wu, and S. Bressan. A framework for conditioning uncertain relational data. In *Proc. DEXA*, 2012.
18. M. van Keulen, A. de Keijzer, and W. Alink. A probabilistic XML approach to data integration. In *Proc. ICDE*, 2005.