THE NATIONAL UNIVERSITY
of SINGAPORE

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

TRB1/12

A Framework for Conditioning Uncertain Relational Data

Ruiming Tang, Reynold Cheng, Huayu Wu and Stephane Bressan.

January 2012

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.

OOI Beng Chin
Dean of School

A Framework for Conditioning Uncertain Relational Data

Ruiming Tang[1], Reynold Cheng[2], Huayu Wu[3], and Stéphane Bressan[4]

[1]School of Computing, National University of Singapore
tangruiming@comp.nus.edu.sg
[2]Department of Computer Science, The University of Hong Kong
ckcheng@cs.hku.hk
[3]Institute for Infocomm Research, Singapore
huwu@i2r.a-star.edu.sg
[4]Center for Maritime Studies
steph@nus.edu.sg

Abstract. We propose a framework for representing conditioned probabilistic relational data. In this framework the existence of tuples in possible worlds is determined by Boolean expressions composed from elementary events. The probability of a possible world is computed from the probabilities associated with these elementary events. In addition, a set of global constraints conditions the database. Conditioning is the formalization of the process of adding knowledge to a database. Some worlds may be impossible given the constraints and the probabilities of possible worlds are accordingly re-defined. The new constraints can come from the observation of the existence or non-existence of a tuple, from the knowledge of a specific rule, such as the existence of an exclusive set of tuples, or from the knowledge of a general rule, such as a functional dependency. We are therefore interested in computing a concise representation of the possible worlds and their respective probabilities after the addition of new constraints, namely an equivalent probabilistic database instance without constraints after conditioning. We devise and present a general algorithm for this computation. Unfortunately, the general problem involves the simplification of general Boolean expressions and is NP-hard. We therefore identify specific practical families of constraints for which we devise and present efficient algorithms.

1 Introduction

The exponential growth of the volume of digital data available for analysis is both a blessing and a curse. It is a blessing for unprecedented applications can be devised that exploit this wealth. It is a curse for the challenges paused by the integration and cleaning of these data are obstacles to their effective usage.

We consider such challenges as modeling, integrating and cleaning economic and scientific data ([4, 9]). We consider the collected data is uncertain and the adjunction of knowledge and observations as a "conditioning process".

Traditional data models, under the strict closed world assumption, presume the exactitude and certainty of the information stored. Modern applications,

such as the ones we consider here, need to process uncertain data coming from diverse and autonomous sources (we do not discuss here the orthogonal issue of the source of uncertainty). This requirement compels new data models able to manage uncertainty. Probabilistic data models are candidates for such solutions. A probabilistic database instance defines a collection of database instances, called *possible worlds*, in an underlying traditional data model: relational, semistructured or else. Each possible world is associated with a probability. The probability may be interpreted as the chance of the instance to be an actual one.

The probabilities of possible worlds are derived from probabilities associated with data units, which can be tuples, attribute values in relational databases, and elements in XML databases, depending on the underlying probabilistic data model. Each probability value can be interpreted as the probability of the data unit to belong to the actual instance. A language of probabilistic events (*events* for short) and expressions can be used to define dependencies among data units.

The addition of knowledge to a probabilistic database instance is called *conditioning*. Conditioning removes possible worlds that do not satisfy the given constraint. Conditioning possibly reduces uncertainty as it may preclude some worlds.

The computation of the remaining possible worlds, of their probabilities and of the respective probabilities of tuples after conditioning is one of the practical challenges.

(1) In this paper, we propose a model for conditioned probabilistic relational databases. For the sake of rigors and generality, we devise a model that natively caters for constraints rather than treating them as add-ons. We devise a tuple-level model as we consider tuples to be the units of observation. We prove that, for every *consistent* conditioned probabilistic database, there exists an *equivalent* probabilistic database on the same sample space without constraints. We define the necessary notions of consistency and equivalence. **(2) We give a general algorithm for the computation of the new equivalent probabilistic database in the general case**. This computation can leverage Bayes' Theorem since the sample space is unchanged. Unfortunately, the general problem is NP-hard. Fortunately, there are some valuable and practical classes of constraints for which we can devise efficient algorithms. **(3) In this paper, we identify two such classes, observation constraints and X-tuple constraints, and present the corresponding efficient algorithms**. An observation constraint is the knowledge of the truth value of an event. In the simplest case, it is the knowledge of the existence or non-existence of a tuple. An X-tuple constraint is the knowledge of the mutual exclusiveness of the existence of some tuples. There are two cases of X-tuple constraints: X-tuple with "maybe" semantics that considers that none of the candidate tuples could exist and X-tuple without "maybe" semantics that requires exactly one of the candidate tuples to exist. Observation constraints and X-tuple constraints naturally stem from integration, lineage and data cleaning applications. The usage of these classes of constraints in such applications are discussed in [1] and [4], for instance.

Let us consider an example of a probabilistic database of port names and their container throughput as collected from various websites. Table 1[1] contains excerpts of information about the container throughput of ports (e.g., the port of Hong Kong). The uncertainty [2] of the collected information is represented by the Boolean expressions and probabilities associated with the Boolean variables in these expressions. The Boolean expression associated with a tuple, as shown in column `exp` of Table 1, is interpreted as the condition for the tuple to exist in the actual instance of the database. The probability associated with a Boolean variable, as shown in Table 2, is to be interpreted as the probability of the variable to be true. It induces the probability of a tuple to exist in the actual instance of the database.

tid	port	year	throughput	exp
t_1	Hong Kong	2007	23.998 million TEUs	e_1
t_2	Hong Kong	2008	24.494 million TEUs	e_2
t_3	Hong Kong	2009	21.040 million TEUs	e_3
t_4	Hong Kong	2009	20.9 million TEUs	e_4

Table 1. Table R

Boolean event	probability
e_1	1/2
e_2	3/5
e_3	1/2
e_4	3/5

Table 2. probability of variables

Let us use the additional knowledge that the container throughput values are unique in the same year of the same port. Conditioning the probabilistic database with this constraint should have the effect to filter those possible worlds that do not satisfy this constraint. Table 3 illustrates four possible worlds that do not satisfy this constraint. The conditioned probabilistic database is obtained by removing these four possible worlds.

tid	port	year	throughput
t_3	Hong Kong	2009	21.040 million TEUs
t_4	Hong Kong	2009	20.9 million TEUs

tid	port	year	throughput
t_1	Hong Kong	2007	23.998 million TEUs
t_3	Hong Kong	2009	21.040 million TEUs
t_4	Hong Kong	2009	20.9 million TEUs

tid	port	year	throughput
t_2	Hong Kong	2008	24.494 million TEUs
t_3	Hong Kong	2009	21.040 million TEUs
t_4	Hong Kong	2009	20.9 million TEUs

tid	port	year	throughput
t_1	Hong Kong	2007	23.998 million TEUs
t_2	Hong Kong	2008	24.494 million TEUs
t_3	Hong Kong	2009	21.040 million TEUs
t_4	Hong Kong	2009	20.9 million TEUs

Table 3. Four possible worlds which do not satisfy the constraint

It is obviously wrong to compute the probabilities by counting the remaining possible worlds as they add up to $1 - \frac{3}{10} = \frac{7}{10}$ ($\frac{3}{10}$ is the sum of probabilities of the four removed possible worlds) and not to 1.

The probabilities of the remaining possible worlds in the conditioned database instance are conditional probabilities in the same sample space. They are com-

[1] TEU stands for Twenty-foot Equivalent Unit. It is a measure of container capacity.

[2] Uncertainty could be quantified in various ways, for instance using ranks, reliability and trust or simply multiplicity of the sources. This issue is orthogonal to the contributions of this paper and is not further discussed.

puted using Bayes' Theorem. Let pwd denote a possible world of the original probabilistic database. Let $p(pwd)$ denote the probability of the possible world pwd in the original probabilistic database. Let $p'(pwd)$ denote the probability of the possible world pwd in the conditioned probabilistic database. The Bayesian equation tells us that $p(pwd \wedge C) = p(pwd|C) \times p(C)$. We know that $p'(pwd) = p(pwd|C)$. Therefore $p'(pwd) = \frac{p(pwd \wedge C)}{p(C)}$. We can now compute the respective probabilities of the remaining possible worlds.

In the example, let us consider the remaining possible world in which t_1, t_2 and t_3 exist. The probability of this possible world in the original database is $p(pwd) = p(e_1) \times p(e_2) \times p(e_3) \times (1 - p(e_4)) = \frac{3}{50}$. The probability of this possible world in the conditioned database is $p'(pwd) = \frac{p(pwd \wedge C)}{p(C)} = \frac{\frac{3}{50}}{\frac{7}{10}} = \frac{3}{35}$.

In the example, an observation constraint that ascertains the existence of the tuple t_1 states that $e_1 = true$. The authors of [4] define cleaning as the addition of such observation constraints.

In the example, an X-tuple without "maybe" semantics constraint that indicates that only one of t_3, t_4 exists, namely that t_3, t_4 form one X-tuple, states that $(e_3 \wedge \neg e_4) \vee (\neg e_3 \wedge e_4)$.

The rest of this paper is organized as follows. Section 2 presents a rapid overview of related work on probabilistic data models and their applications. In Section 3, we present our probabilistic data model and define the conditioning problem. In Section 4, we give general solutions and present practical instances of the general problem for which efficient solution exists and devise such solutions, namely, the observation and X-tuple constraints. We evaluate the performance of the algorithms proposed in Section 5. Finally, we conclude our work in Section 6.

2 Related Work

2.1 Probabilistic Relational Models

A probabilistic database instance represents a collection of database instances, called possible worlds, in an underlying traditional data model. Each possible world is associated with a probability generally derived from probabilities associated with data units of the model. Probabilistic relational data models associate probabilities with relations, with tuples, or with attribute values. Probabilistic XML database models associate probabilities with elements or with values. Probabilistic database models may be limited to independent probabilities associated with individual data units or allow the representation of dependencies.

We are interested in probabilistic relational models. We are not aware of relation-level models. Such models may indeed not be practical. The authors of [8, 12, 6, 1] propose or use tuple-level models. The authors of [2, 11] propose or use attribute-level models. The choice of a model at the appropriate granularity depends on the application.

The model of [12] is the first tuple-level probabilistic relational model. In this model the probability of a tuple is the joint probability of its attributes. Tuples in one probabilistic instance are mutually exclusive and the sum of their probabilities is one. This model is suitable when an instance stores a unique object.

It requires several instances for several objects. The model of [6] overcomes this limitation by using keys to differentiate different objects in the same instance. Objects need not exist in some possible world and therefore the sum of the corresponding probabilities may be less than 1. The model is similar to the model of [1]. A group of tuples representing one object is called an "*X-tuple*". We later refer to the corresponding dependencies as an X-tuple constraint.

The model of [8] associates each tuple with a basic event or a Boolean expression. Events are associated with probabilities. The sample space is the space of events that are true. Possible worlds and their probabilities are defined by the probabilities of events and event expressions to be true. The authors of [7] presents an algebra for this model. The authors of [5, 14] study query evaluation in this model. The authors of [10, 13] study lineage processing in the same model.

The model in [3] is the first proposed attribute-level probabilistic relational model. Attribute values are associated with probabilities. The authors of [2] propose a 1^{st} Normal Form representation of the attribute-level probabilistic relations themselves (by means of a binary model).

As we have seen, existing work on probabilistic relational data model are either tuple-level models or attribute-level models. We propose a tuple-level model with additional constraints integrated as first class citizens. The model we present caters for constraints rather than treating them as add-ons, as other tuple-level models can be adapted to (e.g. [8]).

2.2 Conditioning

The authors of [11] propose an approach to conditioning probabilistic relation instances. They use the attribute-level model of [2]. They adapt algorithms and heuristics for Boolean validity checking and simplification to solve the general NP-hard conditioning problem. As the authors claim, [11] seems to be the only existing work on conditioning probabilistic relational databases.

Our work differentiates itself from [11] in three aspects.

Firstly, although the model we are proposing is fit to tackle the "conditioning problem" as defined in [11], it can also be used to address other problems. This is possible because we have defined constraints as first class citizens of the model. For instance, we can address problems such as the equivalence of conditioned databases or the reverse problem of finding a conditioned database from a given set of possible worlds.

Secondly, we do not only study the conditioning problem in the general case (for which it is proved to be an NP-hard problem) but we also identify some practical classes of constraints for which we can devise efficient algorithms.

Lastly, they do conditioning on an attribute-level model. We propose a tuple-level model for the sake of applications that we are interested in.

3 Probabilistic Data Model

3.1 Probabilistic Relation

A database instance in our probabilistic relational model is a set of database instances in the underlying traditional relational model. A probabilistic database

instance is a collection of probabilistic relation instances. A probabilistic relation instance represents a set of traditional relation instances, called possible worlds, together with a probability for each of these instances to be actual. A probabilistic relation instance is represented as a set of tuples (a traditional relation instance), each of which is associated with a Boolean expression whose truth value determines the presence or absence of the tuple in the actual instance and whose probability to be true or false is defined by probabilities associated with the events of which it is composed. In addition, a set of complex events define integrity *constraints* that instances must verify.

Definition 1. *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.*

$$ce = e|ce \vee ce|ce \wedge ce|ce \rightarrow ce|\neg ce$$

$C(E)$ is the set of complex events formed with the events in E.

Definition 2. *A probabilistic relation instance is a quintuple $< R, E, f, C, p >$ as follows, where R is a traditional relation instance, E is a set of events, f is a function from R to CE, C is a subset of $C(E)$, and p is a function from E to $[0, 1]$.*

In $< R, E, f, C, p >$, E is the set of events used in the probabilistic relation, f is a mapping function to associate each tuple in R with an expression of events, p is a mapping function to specify the probability value of each event being true.

Definition 3. *An* interpretation *of a complex event is a function from E to $\{true, false\}$.*

Definition 4. *The* probability of a complex event *c, noted $p(c)$, is as follows.*

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

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

Figure 1 presents the uncertain information about port mentioned in Section 1, in our probabilistic relation model. R (the first part of the probabilistic relation $\mathcal{R}$ in Figure 1) is the one in Table 1, but without `exp` column.

$$\mathcal{R} = \begin{cases} R \\ E = \{e_1, e_2, e_3, e_4\} \\ f(t_1) = e_1, f(t_2) = e_2, f(t_3) = e_3, f(t_4) = e_4 \\ C = \emptyset \\ p(e_1) = \frac{1}{2}, p_1(e_2) = \frac{3}{5}, p(e_3) = \frac{1}{2}, p(e_4) = \frac{3}{5} \end{cases}$$

Fig. 1. Example of a conditioned probabilistic relation instance

3.2 Possible Worlds

Informally and in short, a possible world is a traditional relation instance such that the complex events associated with the tuples in the possible world are true, the complex event associated with the tuples not in the possible world are false, and the constraints 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 5. *Let $\mathcal{R} =< R, E, f, C, p >$ be a probabilistic relation instance. R' is a possible world of $\mathcal{R}$ if and only if there exists a model of the following formula F with a non zero probability.*

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

where $\mathcal{M}(R')$ is the set of models of the above formula.

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

Note that a possible world R' is a subset of R. Note that R' can be empty.

Definition 6. *Let $\mathcal{R} =< R, E, f, C, p >$ be a probabilistic relation instance. We say that $\mathcal{R}$ is **consistent** (resp. **inconsistent**) if and only if there exists a possible world (resp. there does not exist a possible world) of $\mathcal{R}$, i.e. $\mathfrak{P}(\mathcal{R}) \neq \emptyset$ (resp. $\mathfrak{P}(\mathcal{R}) = \emptyset$).*

Theorem 1. *Let $\mathcal{R} =< R, E, f, C, p >$ be a probabilistic relation instance. $\mathcal{R}$ is inconsistent iff C is always evaluated to be false, i.e. $\mathcal{M}(C) = \emptyset$, where $\mathcal{M}(C)$ is the set of models of C.*

Proof. The sketch of the proof:

1. According to Definition 5, if C is always false, then formula F is always false, which means there does not exist any possible world. Then $\mathcal{R}$ is inconsistent.
2. If $\mathcal{R}$ is inconsistent, there is no possible world. According to Definition 5, F is always false. Suppose C is not always false, we can always find a model of F (possibly generate the possible world with empty content). There is a contradiction. Thus C is always false in this case. □

Definition 7. *Let $\mathcal{R} =< R, E, f, C, p >$ be a probabilistic relation instance. The probability of a tuple $t \in R$, noted $p_{\mathcal{R}}(t)$ is defined as follows.*

$$p_{\mathcal{R}}(t) = \sum_{R' \in \mathfrak{P}(\mathcal{R}) \wedge t \in R'} p_{\mathcal{R}}(R')$$

Theorem 2. *Let $\mathcal{R}_1 =< R, E, f, C, p >$ and $\mathcal{R}_2 =< R, E, f, \emptyset, p >$ be two probabilistic relation instances. If R' is a possible world of $\mathcal{R}_1$ then it is a possible world of $\mathcal{R}_2$.*

Proof. We prove that there exists an interpretation and a non-zero probability.

1. Every model of F_1 below is a model of F_2 below.

$$F_1 = (\bigwedge_{t_i \in R'} f(t_i)) \wedge (\bigwedge_{t_i \notin R'} \neg f(t_i)) \wedge (\bigwedge_{c \in C} c)$$

$$F_2 = (\bigwedge_{t_i \in R'} f(t_i)) \wedge (\bigwedge_{t_i \notin R'} \neg f(t_i))$$

2. Whenever $p_{\mathcal{R}_1}(R') \neq 0$ then $p_{\mathcal{R}_2}(R') \neq 0$.
$p_{\mathcal{R}_1}(R') = p(F_1|C) = \frac{p(F_1 \wedge C)}{p(C)} = \frac{p(F_1)}{p(C)} = \frac{p(F_2 \wedge C)}{p(C)}$, $p_{\mathcal{R}_2}(R') = p(F_2|\emptyset) = p(F_2)$. Whenever $p_{\mathcal{R}_1}(R') \neq 0$, one can get $p(F_2 \wedge C) \neq 0$, then $p(F_2) \neq 0$, thus $p_{\mathcal{R}_2}(R') \neq 0$. □

3.3 World Equivalence

We now define an equivalent relation on one probabilistic relation instance. Two probabilistic relation instances are *world equivalent* if they have the same possible worlds with the same probabilities.

Definition 8. *Let $\mathcal{R}_1 =< R, E_1, f_1, C_1, p_1 >$ and $\mathcal{R}_2 =< R, E_2, f_2, C_2, p_2 >$ be two probabilistic relation instances. We say that $\mathcal{R}_1$ and $\mathcal{R}_2$ are **world equivalent** if and only if:*

$$\forall R' \subset R\ ((R' \in \mathfrak{P}(\mathcal{R}_1)) \leftrightarrow (R' \in \mathfrak{P}(\mathcal{R}_2))) \qquad \textit{and}$$

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

We write $\mathcal{R}_1 \equiv_w \mathcal{R}_2$.

Theorem 3. *Let $\mathcal{R}_1 =< R, E_1, f_1, C, p_1 >$ be a probabilistic relation instance. If $\mathcal{R}_1$ is consistent then there exists a probabilistic relation $\mathcal{R}_2 =< R, E_2, f_2, \emptyset, p_2 >$ such that $\mathcal{R}_2 \equiv_w \mathcal{R}_1$.*

Proof. This is a proof by construction.

For each tuple $t_i \in \mathcal{R}_1$, take the completed disjunctive normal form of $f_1(t_i) \wedge C$. Use d_{ij} to represent each conjunct of $f_1(t_i) \wedge C$, i.e. $f_1(t_i) \wedge C = \bigvee d_{ij}$.

For each distinct d_{ij}, create a new event e_k, using a mapping function(i.e. $r(d_{ij}) = e_k$), and associate e_k with the probability $p_2(e_k) = \frac{p(d_{ij})}{p(C)}$.

We construct mapping function f_2 for $\mathcal{R}_2$ as: $f_2(t_i) = \bigvee_{\bigvee d_{ij} = f_1(t_i) \wedge C, e_k = r(d_{ij})} e_k$. f_2 makes sure $\mathcal{R}_2$ has the same set of possible worlds as $\mathcal{R}_1$.

Based on the constructing f_2, $p(f_2(t_i)) = \frac{p(\bigvee d_{ij})}{p(C)} = \frac{p(f_1(t_i) \wedge C)}{p(C)} = p(f_1(t_i)|C)$. This equation guarantees that $\mathcal{R}_2$ and $\mathcal{R}_1$ associate the same possible world with the same probability value.

Note that the new created events e_k are mutually exclusive. We can use a set of independent events w_k to represent e_k, as: $e_1 = w_1, e_k = \neg w_1 \wedge ... \wedge \neg w_{k-1} \wedge w_k (k = 2, ..., n-1), e_n = \neg w_1 \wedge ... \wedge \neg w_{n-1}$. One can easily compute the probability $p_2(w_k)$ by knowing $p_2(e_k)$. We can also easily replace e_k by independent events w_k, in f_2.

By the construction above, we get $\mathcal{R}_2$ such that $\mathcal{R}_2 \equiv_w \mathcal{R}_1$.

Note that this construction creates a potentially exponential number of variables. Also note that computing probability of general constraints $p(C)$ is an NP-hard problem. □

3.4 Conditioning and the Conditioning Problem

Conditioning a probabilistic relation consists in adding constraints. For example conditioning $\mathcal{R}_1 =< R, E_1, f_1, C_1, p_1 >$ with a set of constraints C is creating the probabilistic relation instance $\mathcal{R}_C =< R, E_1, f_1, C_1 \cup C, p_1 >$. Notice that R, E_1, f_1 and p_1 being unchanged, the underlying sample space is the same.

The conditioning problem consists in finding a world equivalent probabilistic relation instance with no constraint, given one probabilistic relation instance with constraints. Following the example above, given $\mathcal{R}_C =< R, E_1, f_1, C_1 \cup C, p_1 >$, the conditioning problem is finding $\mathcal{R}_2 =< R, E_2, f_2, \emptyset, p_2 >$ such that $\mathcal{R}_2 \equiv_w \mathcal{R}_C$. Notice that R is unchanged.

The existence and form of an equivalent probabilistic relation with no constraint is a direct consequence of Theorem 3, provided the conditioning yields a consistent instance.

Theorem 4. *Let $\mathcal{R}_1 =< R, E_1, f_1, C_1, p_1 >$ be a probabilistic relation instance. Let C be a set of constraints. $\mathcal{R}_C =< R, E_1, f_1, C_1 \cup C, p_1 >$ is the result of conditioning $\mathcal{R}_1$ with C. If $\mathcal{R}_C$ is consistent then the conditioning problem given $\mathcal{R}_C$ always has a solution (i.e. there exists $\mathcal{R}_2 =< R, E_2, f_2, \emptyset, p_2 >$ such that $\mathcal{R}_2 \equiv_w \mathcal{R}_C$).*

4 Algorithms

In this section, we propose different conditioning algorithms for different cases, including the general case of constraints and two special classes of constraints, i.e., observation constraints and X-tuple constraints. We will show that under the special cases, the complexity of solving the conditioning problem can be linear in the size of the probabilistic database, in contrast with NP-hard of the conditioning problem of the general case. For each case, before enforcing constraints, we need to make sure that the conditioned database is consistent. We start from the general case.

4.1 General case

In the general case, the complex events associated with tuples, as well as the constraint C, are general elements of $C(E)$. In order words, there are no other requirements on f values and C.

Checking consistency According to Theorem 1, checking consistency of the conditioned database is to check whether there exists a model of its constraint C. In the general case, the constraint C can be an arbitrary logical expression. Thus actually it is the well known SAT (satisfiability) problem, which is an NP-complete problem.

Conditioning Algorithm 1 is the algorithm solving the conditioning problem with a general constraint, and is adapted from [11]. The general idea of Algorithm 1 follows the construction in the proof of Theorem 3. Algorithm 1 computes the possible worlds (of the conditioned probabilistic database) in which each tuple exists, in the phase 1 (line 1 to line 9). The algorithm computes f_2 and p_2 values using Bayesian equation, in the phase 2 (line 10 to line 17). In this phase, it provides some optimizations to minimize the number of events used during the conditioning process, thus it avoids to compute probabilities of unnecessary events. However, since the conditioning problem is an NP-hard problem in general, no matter how to optimize the algorithm, it is not practical to make it scalable to database size theoretically.

Algorithm 1: Conditioning problem algorithm with a general constraint

Data: $\mathcal{R}_1 =< R, E_1, f_1, C, p_1 >$(C is a general constraint)
Result: $\mathcal{R}_2 =< R, E_2, f_2, \emptyset, p_2 >$ such that $\mathcal{R}_2 \equiv_w \mathcal{R}_1$

1 Phase 1: encoding;
2 **for** *each tuple* $t_i \in R$ **do**
3 | normalize $f_1(t_i)$ to its completed disjunctive normal form;
4 | construct a new R_{new} by duplicating t_i, with its f_1 value being one conjunct clause;
5 normalize C to the completed disjunctive normal form;
6 construct $\mathfrak{C}$ by adding all the conjunct clauses of C;
7 **for** *each* $t_i \in R_{new}$ **do**
8 | **if** $f_1(t_i) \notin \mathfrak{C}$ **then**
9 | | remove t_i;
10 Phase 2: computing f_2 and p_2;
11 **for** *each* $t_i \in R_{new}$ **do**
12 | $f_2(t_i) = update(f_1(t_i))$;
13 | **for** *each event* e'_i *in* $f_2(t_i)$ **do**
14 | | $p_2(e'_i) = p(e_i|(e_1 \wedge ... \wedge e_{i-1} \wedge C))$;
15 | | $//e_i$ is the original basic event of e'_i before updating;
16 **if** *any two* e'_i, e'_j *have the same original event* $\wedge\ p_2(e'_i) = p_2(e'_j)$ **then**
17 | eliminate one of them, replace the eliminated event anywhere by the kept one;

tuple_id	port	year	throughput
t_{3_1}	Hong Kong	2009	21.040 million TEUs
t_{3_2}	Hong Kong	2009	21.040 million TEUs
t_{4_1}	Hong Kong	2009	20.9 million TEUs
t_{4_2}	Hong Kong	2009	20.9 million TEUs

Table 4. The relation R_{new}

A running example Assume that we have one probabilistic relation $\mathcal{R}$ (shown in Section 3), with a general constraint $C = (\neg e_1 \wedge \neg e_2 \wedge e_3) \vee (\neg e_1 \wedge \neg e_2 \wedge e_4)$. The equivalent probabilistic relation without constraints after conditioning should be $< R, E_2, f_2, \varnothing, p_2 >$. After applying encoding phase, the probabilistic relation becomes R_{new}, shown in Table 4. The f_1 values become: $f_1(t_{3_1}) = \neg e_1 \wedge \neg e_2 \wedge$

$e_3 \wedge \neg e_4, f_1(t_{3_2}) = \neg e_1 \wedge \neg e_2 \wedge e_3 \wedge e_4, f_1(t_{4_1}) = \neg e_1 \wedge \neg e_2 \wedge e_3 \wedge e_4, f_1(t_{4_2}) = \neg e_1 \wedge \neg e_2 \wedge \neg e_3 \wedge e_4$. After applying updating f_1 process, we have $f_2(t_{3_1}) = \neg e'_1 \wedge \neg e'_2 \wedge e'_3 \wedge \neg e'_4, f_2(t_{3_2}) = \neg e'_1 \wedge \neg e'_2 \wedge e'_3 \wedge e'_4, f_2(t_{4_1}) = \neg e'_1 \wedge \neg e'_2 \wedge e'_3 \wedge e'_4, f_2(t_{4_2}) = \neg e'_1 \wedge \neg e'_2 \wedge \neg e'_3 \wedge e''_4$.

Then we compute p_2 value for each of these new variables:

$$p_2(\neg e'_1) = p(\neg e_1|C) = \frac{p(\neg e_1 \wedge C)}{p(C)} = \frac{p(C)}{p(C)} = 1, p_2(\neg e'_2) = p(\neg e_2|\neg e_1 \wedge C) = 1$$

$$p_2(e'_3) = p(e_3|\neg e_1 \wedge \neg e_2 \wedge C) = \frac{p(e_3 \wedge \neg e_1 \wedge \neg e_2 \wedge C)}{p(C \wedge \neg e_1 \wedge e_2)} = \frac{p(\neg e_1 \wedge \neg e_2 \wedge e_3)}{p(C)} = \frac{5}{8}$$

$$p_2(e'_4) = p(e_4|\neg e_1 \wedge \neg e_2 \wedge \neg e_3 \wedge C) = \frac{3}{5}, p_2(e''_4) = p(e_4|\neg e_1 \wedge \neg e_2 \wedge \neg e_3 \wedge C) = 1$$

4.2 Special case 1: positive and negative observations

A positive (resp. negative) observation is the knowledge of one event being true (resp. false). After building a probabilistic database, one may get some information about the existence of some tuples or the truth value of some events. This kind of observation is used in some research, as [4]. In [4], the authors build a probabilistic database with the information describing prices of items. There are several possible prices for each item, but only one of them is true. Later, they clean this uncertain information by observing the actual price of some items. Thus, their cleaning operations are adding observation constraints on their probabilistic database.

Checking consistency A probabilistic relation with a set of observation constraints can be consistent or inconsistent. A set of observation constraints form a conjunction of positive and negative events, i.e., C=$p_1 \wedge p_2 \wedge ... \wedge p_m \wedge \neg q_1 \wedge \neg q_2 \wedge ... \wedge \neg q_n$. The approach to check consistency is to check whether there exists the positive and negative form of the same event in C. If there exist e_i and $\neg e_i$ in C, C is always evaluated to be false (because C is a conjunction). In this case, the probabilistic relation is inconsistent. Two iterations are needed to check that, thus the time complexity is $O(n^2)$, where n is the number of events in C.

Conditioning Algorithm 2 is the algorithm solving the conditioning problem with a set of observation constraints. The basic idea is to update the probabilities of the positive events in the constraints to 1, and to update the probabilities of the negative events in the constraints to 0. As we can see, the only component updated in Algorithm 2 is p_1, while other parts remain the same. The time complexity of Algorithm 2 is linear in the size of the constraint. In the worst case the time complexity is *O(n)*, where n is the number of tuples in the probabilistic relation. There are two lines (line 2 and line 5) needed only when we compute $p(C)$.

A running example Assume that we have one probabilistic relation $\mathcal{R}$ (shown in Section 3), with a set of observation constraints $C = e_1 \wedge \neg e_3$. The equivalent probabilistic relation without constraints after conditioning should be $< R, E, f, \varnothing, p_2 >$, with $p_2(e_1) = 1, p_2(e_2) = \frac{3}{5}, p_2(e_3) = 0, p_2(e_4) = \frac{3}{5}$.

Algorithm 2: Conditioning problem algorithm with a set of observation constraints

Data: $\mathcal{R}_1 =< R, E_1, f_1, C, p_1 >$(C is a set of observations)
Result: $\mathcal{R}_2 =< R, E_2, f_2, \varnothing, p_2 >$ such that $\mathcal{R}_2 \equiv_w \mathcal{R}_1$

for *each* $e_i \in C$ **do**
  $p(C) \times = p_1(e_i)$;//only needed when computing p(C);
  $p_2(e_i) = 1$;
for *each* $\neg e_i \in C$ **do**
  $p(C) \times = (1 - p_1(e_i))$;//only needed when computing p(C);
  $p_2(e_i) = 0$;

4.3 Special case 2: X-tuple without "maybe" semantics

In this section, for $\mathcal{R} =< R, E, f, C, p >$, we consider the special case that each f value is one unique event.

Definition 9. *An **X-tuple** is a set of tuples among which every tuple is mutually exclusive to others. An **X-tuple** without "maybe" semantics means within the X-tuple, one tuple must be chosen. The independence between different X-Tuples are assumed.*

Many research works (such as [1, 4]) use X-tuple model as their probabilistic data model. Thus, their probabilistic data model can be viewed as the conditioning result of adding a set of X-tuple constraints. This section considers about the X-tuple without "maybe" semantics that requires exactly one of the candidate tuples to exist. The next section considers about the X-tuple with "maybe" semantics that considers none of the candidate tuples could exist.

Assume we have k X-tuples: $X_1, X_2, ..., X_k$, and each X-tuple X_i has n_i candidate tuples: $t_{(i,1)}, ..., t_{(i,n_i)}$. The constraint of an X-tuple X_i (with n_i tuples) without "maybe" semantics can be expressed as:

$$C_i = \bigvee_{j=1,...,n_i} (f(t_{(i,j)}) \wedge (\bigwedge_{r=1,...,n_i \wedge r \neq j} \neg f(t_{(i,r)})))$$

Thus a set of X-tuples without "maybe" semantics constraints telling that there are k X-tuples present as:

$$C = \bigwedge_{i=1,...,k} C_i$$

For example, in the probabilistic relation $\mathcal{R}$ (shown in Section 3), if we have a set of X-tuple without "maybe" semantics constraints telling that there are two X-tuples: $X_1 = \{t_1, t_2\}, X_2 = \{t_3, t_4\}$, the set of constraints C present as:

$$C = ((e_1 \wedge \neg e_2) \vee (\neg e_1 \wedge e_2)) \wedge ((e_3 \wedge \neg e_4) \vee (\neg e_3 \vee e_4))$$

Checking consistency In this special case, one probabilistic relation with a set of X-tuple without "maybe" semantics constraints, with each f value being one unique event, is always consistent. According to Definition 6 and Theorem 1, in order to check consistency, we need to check whether C is always evaluated to be false or not. From the above example, if we transform the constraint C into its disjunctive normal form, we can see that each event involved in C appears only once in each conjunct. Thus the case of $e_i \wedge \neg e_i$ never happens, which gives chances to C to be true. In this case, one probabilistic relation is always consistent.

Conditioning

Definition 10. *A **compact encoding** of an X-tuple without "maybe" semantics X_i, which has n_i tuples, is:*

$$f(t_{(i,j)}) = \begin{cases} e_{i_j}, j{=}1 \\ \neg e_{i_1} \wedge ... \wedge \neg e_{i_{j-1}} \wedge e_{i_j}, j{=}2, \ ..., \ n_i - 1 \\ \neg e_{i_1} \wedge ... \wedge \neg e_{i_{j-1}}, j{=}n_i \end{cases}$$

where $t_{(i,j)}$ represents the j-th tuple of X_i, j=1, ..., n_i, and all e_{i_j} are new events which have never been used.

Theorem 5. *The compact encoding can express the X-tuple semantics.*

Proof.
1. We can see that $t_{(i,1)}$ is mutually exclusive to all other tuples, since $f(t_{(i,1)}) = e_{i_1}$ and the f values of all other tuples contain $\neg e_{i_1}$.
2. We can see that $t_{(i,n_i)}$ is mutually exclusive to all other tuples. If we have another tuple $t_{(i,p)}$, $f(t_{(i,p)}) = \neg e_{i_1} \wedge ... \wedge \neg e_{i_{p-1}} \wedge e_{i_p}$. $f(t_{(i,n_i)}) = \neg e_{i_1} \wedge ... \wedge \neg e_{i_p} ... \wedge \neg e_{i_{n_i-1}}$. e_{i_p} is in $f(t_{(i,p)})$, while $\neg e_{i_p}$ is in $f(t_{(i,n_i)})$. Thus, they are mutually exclusive.
3. If we get any two tuples $t_{(i,p)}$ and $t_{(i,q)}$, $p < q, p, q \in [2, n_i - 1]$, they are mutually exclusive. $f(t_{(i,p)}) = \neg e_{i_1} \wedge ... \wedge \neg e_{i_{p-1}} \wedge e_{i_p}$, $f(t_{(i,q)}) = \neg e_{i_1} \wedge ... \wedge \neg e_{i_p} \wedge ... \wedge \neg e_{i_{q-1}} \wedge e_{i_q}$. e_{i_p} is in $f(t_{(i,p)})$, while $\neg e_{i_p}$ is in $f(t_{(i,q)})$. Thus, they are mutually exclusive. □

The independence is assumed among X-tuples, thus when we compute the probability of one tuple $t_{(i,j)}$ existing in the equivalent probabilistic database without constraints, we only need to consider the X-tuple X_i containing $t_{(i,j)}$.

Theorem 6. *$\mathcal{R}_1 =< R, E_1, f_1, C, p_1 >$ is one probabilistic relation. C is a set of X-tuple without "maybe" semantics constraints claiming that there exist k X-tuples: $X_1, X_2, ..., X_k$, while each X-tuple X_i has n_i candidate tuples: $t_{(i,1)}, ..., t_{(i,n_i)}$. The probability of one tuple $t_{(i,j)}$ existing in the equivalent conditioned probabilistic database without constraints, $\mathcal{R}_2 =< R, E_2, f_2, \varnothing, p_2 >$, can be computed as:*

$$p(f_2(t_{(i,j)})) = \frac{dif_{ij}}{XDif_i}$$

where $dif_{ij} = \frac{p(f_1(t_{(i,j)}))}{1-p(f_1(t_{(i,j)}))}$, and $XDif_i = \sum_{j=1}^{n_i} dif_{ij}$.

Proof. Since independence is assumed among X-tuples, we only need to consider about the sub-constraint C_i which is the constraint claiming X-tuple X_i. According to the semantics of an X-tuple constraint, we can compute the probability of C_i as:

$$\begin{aligned}
p(C_i) &= \sum_{j=1}^{n_i} p(f_1(t_{(i,j)}) \wedge \bigwedge_{r=1,\ldots,n_i \wedge r \neq j} \neg f_1(t_{(i,r)})) \\
&= \sum_{j=1}^{n_i} (p(f_1(t_{(i,j)})) \times \prod_{r=1,\ldots,j-1,j+1,\ldots n_i} (1 - p(f_1(t_{(i,r)})))) \\
&= \sum_{j=1}^{n_i} ((\prod_{r=1}^{n_i} (1 - p(f_1(t_{(i,r)})))) \times \frac{p(f_1(t_{(i,j)}))}{1 - p(f_1(t_{(i,j)}))}) \\
&= (\prod_{r=1}^{n_i} (1 - p(f_1(t_{(i,r)})))) \times XDif_i
\end{aligned}$$

We can easily get:

$$\begin{aligned}
p(f_1(t_{(i,j)}) \wedge C_i) &= (\prod_{r=1}^{n_i} (1 - p(f_1(t_{(i,r)})))) \times \frac{p(f_1(t_{(i,j)}))}{1 - p(f_1(t_{(i,j)}))} \\
&= (\prod_{r=1}^{n_i} (1 - p(f_1(t_{(i,r)})))) \times dif_{ij}
\end{aligned}$$

Thus we can induce:

$$p(f_2(t_{(i,j)})) = p(f_1(t_{(i,j)})|C_i) = \frac{p(f_1(t_{(i,j)}) \wedge C_i)}{p(C_i)} = \frac{dif_{ij}}{XDif_i}$$

□

Using Theorem 6, we can save lots of computations. When computing $p(f_2(t_{(i,j)}))$, we only need to compute dif_{ij} and $XDif_i$ of X-tuple X_i.

Actually, we can also simplify the computation of probability of the constraint C, which is a set of X-tuple constraints, using the theorem below.

Theorem 7. *$\mathcal{R}_1 =< R, E_1, f_1, C, p_1 >$ is one probabilistic relation. C is a set of X-tuple without "maybe" semantics constraints claiming there exist k X-tuples: $X_1, X_2, ..., X_k$, while each X-tuple X_i has n_i candidate tuples: $t_{(i,1)}, ..., t_{(i,n_i)}$. The probability of C can be computed as:*

$$p(C) = Base \times Dif$$

where $Base = \prod_{i=1}^{k} XBase_i$, $XBase_i = \prod_{r=1}^{n_i}(1-p(f_1(t_{(i,r)})))$, $Dif = \prod_{i=1}^{k} XDif_i$. $XDif_i$ is defined in Theorem 6.

Proof. Since X_i, X_j are independent, C_i, C_j are independent as well. We can compute the probability of C, as (using Theorem 6):

$$p(C) = \prod_{i=1}^{k} p(C_i) = \prod_{i=1}^{k}((\prod_{r=1}^{n_i}(1 - p(f_1(t_{(i,r)})))) \times XDif_i)$$
$$= \prod_{i=1}^{k}(Xbase_i \times XDif_i)$$
$$= (\prod_{i=1}^{k} Xbase_i) \times (\prod_{i=1}^{k} XDif_i) = Base \times Dif$$

□

Algorithm 3: Conditioning problem algorithm with a set of X-Tuple without "maybe" semantics constraints

Data: $\mathcal{R}_1 =< R, E_1, f_1, C, p_1 >$(C is a set of X-tuple constraints)
Result: $\mathcal{R}_2 =< R, E_2, f_2, \varnothing, p_2 >$ such that $\mathcal{R}_2 \equiv_w \mathcal{R}_1$

1 Phase 1: encoding;//only needed when doing conditioning;
2 **for** *each X-tuple* X_i **do**
3 **for** *each tuple* $t_{(i,j)}$ *of* X_i **do**
4 update $f_1(t_{(i,j)})$ to $f_2(t_{(i,j)})$ by encoding in a compact manner;
5 Phase 2: computing;
6 **for** *each X-tuple* X_i **do**
7 $XBase_i = 1$; $XDif_i = 0$;
8 **for** *each tuple* t_{ij} *of* X_i **do**
9 $base_{ij} = 1 - p(f_1(t_{(i,j)}))$;$XBase_i \times = base_{ij}$;//only needed when computing p(C);
10 $dif_{ij} = \frac{p(f_1(t_{(i,j)}))}{1-p(f_1(t_{(i,j)}))}$; $XDif_i += dif_{ij}$;
11 $Base = \prod_{i=1}^{k} XBase_i$;//only needed when computing p(C);
12 $Dif = \prod_{i=1}^{k} XDif_i$;
13 $p(C) = Base \times Dif$;
14 **for** *each X-tuple* X_i **do**
15 **for** *each tuple* $t_{(i,j)}$ *of* X_i **do**
16 set up an equation $p(f_2(t_{(i,j)})) = \frac{dif_{ij}}{XDif_i}$;
17 solve the equations, and get the p_2 value for all new variables in f_2;

Algorithm 3 is the algorithm solving the conditioning problem for a set of X-tuple without "maybe" semantics constraints. Algorithm 3 is built based on Theorem 5, 6 and 7. The encoding phase (line 1 to line 4) is based on Theorem 5. The computing phase (line 5 to line 17) comes from Theorem 6 and 7. Actually, we do not need to compute *p(C)* (line 9 and line 11) if we only aim to solve the conditioning problem. However, the probability of C can be computed by Algorithm 3, without affecting the time complexity. Obviously, the complexity of Algorithm 3 is linear in the size of X-tuple constraints. In worst case, it is *O(n)*, where n is the size of the probabilistic relation.

A running example Assume that we have one probabilistic relation $\mathcal{R}$ (shown in Section 3), with a set of X-tuple constraints $C = ((e_1 \wedge \neg e_2) \vee (\neg e_1 \wedge e_2)) \wedge ((e_3 \wedge \neg e_4) \vee (\neg e_3 \vee e_4))$ (it tells t_1, t_2 is one X-tuple, and t_3, t_4 is another X-tuple). The equivalent probabilistic relation without constraints after conditioning should be $< R, E_2, f_2, \varnothing, p_2 >$

According to the compact encoding, we have f_2 function:

$$f_2(t_1) = e'_1, f_2(t_2) = \neg e'_1, f_2(t_3) = e'_2, f_2(t_4) = \neg e'_2,$$

and $E_2 = \{e'_1, e'_2\}$.

Next, compute the p_2 function.

$$Base = (1 - p_1(e_1)) \times (1 - p_1(e_2)) \times (1 - p_1(e_3)) \times (1 - p_1(e_4)) = \frac{1}{25}$$

$$XDif_1 = \frac{p_1(e_1)}{1 - p_1(e_1)} + \frac{p_1(e_2)}{1 - p_1(e_2)} = \frac{5}{2}, XDif_2 = \frac{p_1(e_3)}{1 - p_1(e_3)} + \frac{p_1(e_4)}{1 - p_1(e_4)} = \frac{5}{2}$$

$$XDif = XDif_1 \times XDif_2 = \frac{25}{4}$$

$$p(C) = Base \times XDif = \frac{1}{25} \times \frac{25}{4} = \frac{1}{4}$$

$$p(f_2(t_1)) = \frac{\frac{p_1(e_1)}{1-p_1(e_1)}}{\frac{p_1(e_1)}{1-p_1(e_1)} + \frac{p_1(e_2)}{1-p_1(e_2)}} = \frac{2}{5}, p(f_2(t_2)) = \frac{\frac{p_1(e_2)}{1-p_1(e_2)}}{\frac{p_1(e_1)}{1-p_1(e_1)} + \frac{p_1(e_2)}{1-p_1(e_2)}} = \frac{3}{5}$$

$$p(f_2(t_3)) = \frac{\frac{p_1(e_3)}{1-p_1(e_3)}}{\frac{p_1(e_3)}{1-p_1(e_3)} + \frac{p_1(e_4)}{1-p_1(e_4)}} = \frac{2}{5}, p(f_2(t_4)) = \frac{\frac{p_1(e_4)}{1-p_1(e_4)}}{\frac{p_1(e_3)}{1-p_1(e_3)} + \frac{p_1(e_4)}{1-p_1(e_4)}} = \frac{3}{5}$$

Thus we have p_2 function:

$$p_2(e'_1) = \frac{2}{5}, p_2(e'_2) = \frac{2}{5}.$$

4.4 Special case 3: X-tuple with "maybe" semantics

In this section, we study the case of X-tuple with "maybe" semantics constraint that considers none of the candidate tuples could exist. X-tuple with"maybe" semantics has a slightly different compact encoding approach.

Assume we have k X-tuples: $X_1, X_2, ..., X_k$, and each X-tuple X_i has n_i candidate tuples: $t_{(i,1)}, ..., t_{(i,n_i)}$. The constraint of an X-tuple X_i (with n_i tuples) with "maybe" semantics can be expressed as:

$$C_i = (\bigvee_{j=1,...,n_i} (f(t_{(i,j)}) \wedge (\bigwedge_{r=1,...,n_i \wedge r \neq j} \neg f(t_{(i,r)})))) \vee (\wedge_{r=1,...,n_i} \neg f(t_{(i,r)}))$$

Thus a set of X-tuples with "maybe" semantics constraints telling that there are k X-tuples present as:

$$C = \bigwedge_{i=1,...,k} C_i$$

Definition 11. *A **compact encoding** of an X-tuple with "maybe" semantics X_i, which has n_i tuples, is:*

$$f(t_{(i,j)}) = \begin{cases} e_{i_j}, j{=}1 \\ \neg e_{i_1} \wedge ... \wedge \neg e_{i_{j-1}} \wedge e_{i_j}, j{=}2, ..., n_i \end{cases}$$

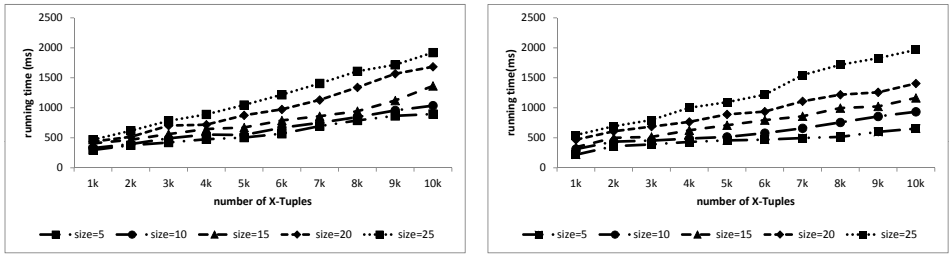
where $t_{(i,j)}$ represents the j-th tuple of X_i, j=1, ..., n_i.

The algorithm solving the conditioning problem for a set of X-tuple with "maybe" semantics constraints is almost the same as Algorithm 3, except that we need to insert one line between line 10 and 11: "$XDif_i+ = 1$;". Thus we omit the pseudo-code here.

5 Performance Evaluation

We mainly show the advantage of our approach in handling special classes of constraints, i.e., achieving scalable performance. Since the first special case is trivial, we only test the performance of the last two special cases, which are X-tuple without and with "maybe" semantics constraints.

All algorithms were implemented in Java and performed by a 2.83GHz CPU with 3.00GB RAM. We generated $300k$ tuples as our synthetic data set. f value of each tuple is one unique event, and the probability of each event is randomly assigned between [0,1]. Every X-tuple contains the same number of candidate tuples. We use two parameters to control the complexity of X-tuple constraint: the number of X-tuples and the number of candidate tuples in each X-tuple.



(a) X-tuple without "maybe" semantics (b) X-tuple with "maybe" semantics

Fig. 2. Running time varying number of X-tuples and size of X-Tuples

Figure.2 shows the running time by differentiating the cases of X-tuples with different number of candidate tuples, and varying the number of X-tuples. Figure.2(a) and Figure.2(b) show the running time of solving the conditioning problem with a set of X-tuple without and with "maybe" semantics constraints respectively. The number of X-tuples varies from $1k$ to $10k$. We performed five runs, each of which adopts different number of candidate tuples in X-tuples, varying from 5 to 25.

We can observe that for both cases, when we fix the number of X-tuples, the running time increases as the number of candidate tuples in each X-tuple increases. When we fix the number of candidate tuples in each X-tuple, the

running time increases as the number of X-tuples increases. The reason is that when the number of X-tuples or the number of candidate tuples in each X-tuple increases, it takes more effort to compute the probability of the constraint, and to conditioning the probabilistic relation.

Furthermore, the time of each run is scalable to the number of X-tuple, and thus scalable to the relation size. It verifies our theoretical analysis of the time complexity of the algorithm for the special class of X-tuple constraints.

6 Conclusion

In this paper, we formalize a tuple-level probabilistic relational data model with additional constraints integrated as first class citizens. The model gives the flexibility to incorporate knowledge and observations in the form of constraints which are specified at any time. This process is called "conditioning". Furthermore, we propose and prove results and devise algorithms for the compact representation and the computation of possible worlds and their probabilities after conditioning. We also identify valuable and practical classes of constraints (i.e., observation constraints and X-tuple constraints), devise efficient algorithms for them, and demonstrate that solving the conditioning problem under them can be performed in linear time.

References

1. 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.
2. L. Antova, T. Jansen, C. Koch, and D. Olteanu. Fast and simple relational processing of uncertain data. In *ICDE*, pages 983–992, 2008.
3. D. Barbará, H. Garcia-Molina, and D. Porter. The management of probabilistic data. *IEEE Trans. on Knowl. and Data Eng.*, 4:487–502, October 1992.
4. R. Cheng, J. Chen, and X. Xie. Cleaning uncertain data with quality guarantees. *PVLDB*, 1(1):722–735, 2008.
5. N. N. Dalvi and D. Suciu. Efficient query evaluation on probabilistic databases. In *VLDB*, pages 864–875, 2004.
6. D. Dey and S. Sarkar. Psql: A query language for probabilistic relational data. *Data Knowl. Eng.*, 28(1):107–120, 1998.
7. R. Fink, D. Olteanu, and S. Rath. Providing support for full relational algebra in probabilistic databases. In *ICDE*, pages 315–326, 2011.
8. 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.
9. A. Y. Halevy, A. Rajaraman, and J. J. Ordille. Data integration: The teenage years. In *VLDB*, pages 9–16, 2006.
10. B. Kanagal and A. Deshpande. Lineage processing over correlated probabilistic databases. In *SIGMOD Conference*, pages 675–686, 2010.
11. C. Koch and D. Olteanu. Conditioning probabilistic databases. *PVLDB*, 2008.
12. M. Pittarelli. An algebra for probabilistic databases. *IEEE Trans. Knowl. Data Eng.*, 6(2):293–303, 1994.
13. C. Ré and D. Suciu. Approximate lineage for probabilistic databases. *PVLDB*, 1(1), 2008.
14. M. A. Soliman, I. F. Ilyas, and K. C.-C. Chang. Top-k query processing in uncertain databases. In *ICDE*, pages 896–905, 2007.