

Rule-based Conditioning of Probabilistic Data

M. van Keulen¹, B.L. Kaminski², C. Matheja², and J.-P. Katoen^{1,2}

¹ University of Twente, {m.vankeulen,j.p.katoen}@utwente.nl

² RWTH Aachen, {benjamin.kaminski,matheja,katoen}@cs.rwth-aachen.de

Abstract. Data interoperability is a major issue in data management for data science and big data analytics. Probabilistic data integration (PDI) is a specific kind of data integration where extraction and integration problems such as inconsistency and uncertainty are handled by means of a probabilistic data representation. This allows a data integration process with two phases: (1) a quick partial integration where data quality problems are represented as uncertainty in the resulting integrated data, and (2) using the uncertain data and continuously improving its quality as more evidence is gathered. The main contribution of this paper is an iterative approach for incorporating evidence of users in the probabilistically integrated data. Evidence can be specified as hard or soft rules (i.e., rules that are uncertain themselves).

Keywords: data cleaning, data integration, information extraction, probabilistic databases, probabilistic programming

1 Introduction

Data interoperability is a major issue in data management for data science and big data analytics. It may be hard to extract information from certain kinds of sources (e.g., natural language, websites), it may be unclear which data items should be combined when integrating sources, or they may be inconsistent complicating a unified view, etc. *Probabilistic data integration* (PDI) [1] is a specific kind of data integration where extraction and integration problems such as inconsistency and uncertainty are handled by means of a probabilistic data representation. The approach is based on the view that data quality problems (as they occur in an integration process) can be modeled as uncertainty [3] and this uncertainty is considered an important result of the integration process [4].

The PDI process contains two phases (see Figure 1):

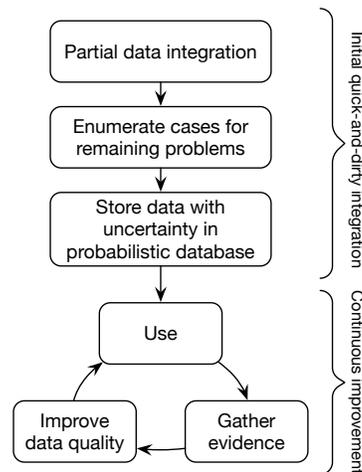


Fig. 1. Probabilistic data integration process [1, 2]

the integration process [4].

- a quick partial integration where certain data quality problems are not solved immediately, but explicitly represented as uncertainty in the resulting integrated data stored in a probabilistic database;
- continuous improvement by using the data — a probabilistic database can be queried directly resulting in possible or approximate answers [5, 6] — and gathering evidence (e.g., user feedback) for improving the data quality.

For details on the first phase, we refer to [2, 3], as well as [7–9] for techniques on specific extraction and integration problems (merging semantic duplicates, merging grouping data, and information extraction from natural language text, respectively). This paper focuses on the second phase of this process, namely on the problem of how to incorporate evidence of users in the probabilistically integrated data with the purpose to continuously improve its quality as more evidence is gathered. We assume that evidence of users is obtained in the form of rules expressing what is necessary (in case of *hard rules*) or likely (in case of *soft rules*) to be true. Rules may focus on individual data items, individual query results, or may state general truths based on background knowledge of the user about the domain at hand. The paper proposes a method for incorporating the knowledge expressed by a rule in the integrated data by means of *conditioning* the probabilistic data on the observation that the rule is true.

In probabilistic programming and statistical relational learning, it is common to answer queries of the form $P(Q|E)$, where E denotes evidence [10, 11], whereas probabilistic databases typically focus on scalable answering of top-k queries without considering evidence [12]. A notable exception is [13] which accounts for “improbable worlds” during query processing. Note that our approach to evidence is fundamentally different: instead of a query mechanism for computing $P(Q|E)$, we incorporate E in the database, such that computing a subsequent $P(Q)$ effectively determines $P(Q|E)$. This allows for an iterative more scalable incorporation of accumulating evidence.

Contributions. This paper makes the following contributions:

- A technique to remap random variables (in this paper referred to as partitionings) to fresh ones in a probabilistic database.
- An extension to probabilistic query languages to specify evidence as hard and soft rules.
- An approach to incorporate such specified evidence in a probabilistic database by updating it.

Outlook. The paper is structured as follows. Section 1.1 presents a running example based on an information extraction scenario. Section 2 gives the background on probabilistic databases, the probabilistic datalog language (JudgeD), and how results from probabilistic data integration can be stored in a probabilistic database. Section 3 describes and explains all contributions, in particular how to rewrite (i.e., update) a probabilistic database with rule evidence into one in which the evidence is incorporated. Section 4 presents a sketch of the main proof: the semantics of a probabilistic database with evidence incorporated in it is equivalent to the semantics of a probabilistic database with its evidence still separate.

1.1 Running example

Throughout the paper we use an information extraction scenario as running example: the “*Paris Hilton example*”. Although this scenario is from the Natural Language Processing (NLP) domain, note that it is equally applicable to other data integration scenarios such as semantic duplicates [7], entity resolution, uncertain groupings [8], etc.

Paris Hilton example. This example and the problem of incorporating rule-based knowledge by means of conditioning was first described in [14]. We summarize it here.

Because natural language is highly ambiguous and computers are still incapable of ‘real’ semantic understanding, information extraction (IE) from natural language is an inherently imperfect process. We focus in this example on the sentence

“Paris Hilton stayed in the Paris Hilton.”

A *named entity* (NE) is a phrase that is to be interpreted as a name referring to some entity in the real world. A specific task in IE is *Named Entity Recognition* (NER): detecting which phrases in a text are named entities, possibly also detecting the type of the NE. The resulting data of this task is typically in the form of *annotations*.

Here we have two NEs which happen to be the same phrase “Paris Hilton”. It is ambiguous how to interpret it: it could be a person, a hotel, or even a fragrance. In fact, we as humans unconsciously understand that the first mention of “Paris Hilton” must refer to a person and the second to a hotel, because from the $3 \times 3 = 9$ combinations only ‘person–stay in–hotel’ seems logical (based on our background knowledge unknown to the IE algorithm).

Often ignored in NER, also the word “Paris” is a NE: it could be a first name or a city. Note that interpretations are correlated: if “Paris” is interpreted as a city, then “Paris Hilton” is more likely to be a hotel, and vice versa. The evidence a user may want to express is

- words contained in phrases interpreted as persons, should not be interpreted as cities, or
- ‘stay-in’ relationships between entities will not have buildings (such as hotels) on the lefthand side and no persons on the righthand side.

In this example, we assume that the initial information extraction produces a *probabilistic database with uncertain annotations* [9, 15]: the type of the first “Paris Hilton” can be either a hotel, person, or fragrance with probabilities 0.5, 0.4, 0.1, respectively. The second “Paris Hilton” analogously. Both mentions of “Paris” are of type firstname or city. The contributions of this paper allow for expressing the evidence in a query language and update the database accordingly resulting in a database with less uncertainty and of higher quality (i.e., closer to the truth).

2 Background

2.1 Probabilistic database

A common foundation for probabilistic databases is possible worlds theory [5]. We follow the formalization of [16] as it separates (a) the data model and the mechanism for handling uncertainty, and (b) the abstract notion of *worlds* and the data contained in them.

Probabilistic database. We view a *database* $DB \in \mathbb{P}A$ as a set of *assertions* $\{a_1, \dots, a_n\}$. For the purpose of data model independence, we abstract from what an assertion is: it may be a tuple in a relational database, a node in an XML database, and so on. A *probabilistic database* $PDB \in \mathbb{P}PA$ is defined as a finite set of possible database states.

Partitionings and descriptive sentences. We postulate an infinite set of *worlds*. An assertion is contained only in a subset of all possible worlds. To describe this relationship, we introduce an identification mechanism, called *descriptive sentence*, to refer to a subset of the possible worlds. If two worlds contain the same assertions, they are said to be *indistinguishable* and we regard them as one possible world. As a consequence, this effectively defines a finite set of distinguishable possible worlds representing the possible database states. We use the symbols DB and w interchangeably.

Let Ω be the set of partitionings. A *partitioning* $\omega^n \in \Omega$ introduces a set of n labels $l \in L(\omega^n)$ of the form $\omega = v$ (without loss of generality, we assume $v \in 1..n$). A partitioning splits the set of possible worlds into n disjunctive subsets $W(l)$. A *descriptive sentence* φ is a propositional formula over the labels. Let $\omega(\varphi)$ be the set of partitionings contained in formula φ . The symbols \top and \perp denote the *true* and *false* sentences. A sentence φ denotes a specific subset of worlds:

$$W(\varphi) = \begin{cases} PDB & \text{if } \varphi = \top \\ \emptyset & \text{if } \varphi = \perp \\ W(l) & \text{if } \varphi = l \\ W(\varphi_1) \cap W(\varphi_2) & \text{if } \varphi = \varphi_1 \wedge \varphi_2 \\ W(\varphi_1) \cup W(\varphi_2) & \text{if } \varphi = \varphi_1 \vee \varphi_2 \\ PDB \setminus W(\varphi_1) & \text{if } \varphi = \neg \varphi_1 \end{cases} \quad (1)$$

A *fully described sentence* $\bar{\varphi}$ over $\Omega = \{\omega_1^{n_1}, \dots, \omega_k^{n_k}\}$ is a formula $\bigwedge_{i \in 1..k} l_i$ with $l_i \in L(\omega_i^{n_i})$. It denotes a set of exactly one world, hence can be used as the *name* or *identifier* for a world. Let $\Phi(\Omega)$ be the set of all fully described sentences over Ω . The following holds:

$$PDB = \bigcup_{\bar{\varphi} \in \Phi(\Omega)} W(\bar{\varphi}) \quad (2)$$

$$PDB = \bigcup_{l \in L(\omega^n)} W(l) \quad (\forall \omega^n \in \Omega) \quad (3)$$

Compact probabilistic database. A compact probabilistic database is a tuple $CPDB = \langle \widehat{DB}, \Omega, P \rangle$ where \widehat{DB} is a set of descriptive assertions $\hat{a} = \langle a, \varphi \rangle$, Ω a set of partitionings, and P a probability assignment function for labels provided that $\sum_{v=1}^n P(\omega^n=v) = 1$. Figure 2 illustrates these notions. We consider $CPDB$ to be *well-formed* if all assertions a used in $CPDB$ occur only once. Well-formedness can always easily be obtained by ‘merging duplicate assertions’ using the transformation rule $\langle a, \varphi_1 \rangle, \langle a, \varphi_2 \rangle \mapsto \langle a, \varphi_1 \vee \varphi_2 \rangle$. We use the terms assertion and data item interchangeably. The possible worlds of $CPDB = \langle \widehat{DB}, \Omega, P \rangle$ are obtained as follows:

$$W(CPDB) = \{DB \mid \bar{\varphi} \in \Phi(\Omega) \wedge DB = \{a \mid \langle a, \varphi \rangle \in \widehat{DB} \wedge \bar{\varphi} \Rightarrow \varphi\}\} \quad (4)$$

This setup naturally supports to express several important dependency relationships:

- *Mutual dependence*: for $\langle a_1, \varphi \rangle$ and $\langle a_2, \varphi \rangle$, either a_1 and a_2 both exist in a world or none of them, but never only one of the two.
- *Mutual exclusivity*: for $\langle a_1, \varphi_1 \rangle$ and $\langle a_2, \varphi_2 \rangle$, it holds that a_1 and a_2 never both occur in a world if $\varphi_1 \wedge \varphi_2 \equiv \perp$.
- *Independence*: Since each ω_i is a partitioning on its own, it can be considered as an independent random variable making an independent choice. For example, $\langle a_1, x=1 \rangle$ and $\langle a_2, y=1 \rangle$ use different partitionings, hence their existence in worlds is independent and a world can contain both a_1 and a_2 , only one of the two, or none of them.

Probability calculation. The probability of a sentence $P(\varphi)$ can be derived from label probabilities making use of properties like $P(\omega_1=v_1 \wedge \omega_2=v_2) = P(\omega_1=v_1) \times P(\omega_2=v_2)$ and $P(\omega_1=v_1 \vee \omega_2=v_2) = P(\omega_1=v_1) + P(\omega_2=v_2)$ if $\omega_1 \neq \omega_2$. The probability of a world is defined as $P(w) = P(\bar{\varphi})$ with $W(\bar{\varphi}) = \{w\}$. The probability of a descriptive assertion is defined as $P(\langle a, \varphi \rangle) = P(\varphi)$. It holds that:

$$P(\langle a, \varphi \rangle) = \sum_{w \in PDB, a \in w} P(w) = \sum_{w \in W(\varphi)} P(w) = P(\varphi)$$

Probabilistic querying. The concept of possible worlds means that querying a probabilistic database should produce the same answer as querying each possible world separately. Given traditional query results $Q(DB)$, let:

$$Q(PDB) = \{Q(DB) \mid DB \in PDB\}$$

As explained in [16], we abstract from specific operators analogously to the way we abstract from the form of the actual data items. Given a query language,

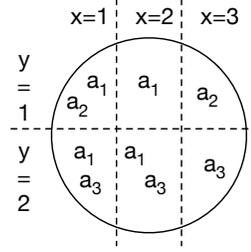


Fig. 2. Illustration of a compact probabilistic database $CPDB = \langle \widehat{DB}, \Omega, P \rangle$. $\widehat{DB} = \{\langle a_1, \neg x=3 \rangle, \langle a_2, \neg x=2 \wedge y=1 \rangle, \langle a_3, y=2 \rangle\}$. $\Omega = \{x^3, y^2\}$. $W(CPDB) = \{\{a_1\}, \{a_2\}, \{a_3\}, \{a_1, a_2\}, \{a_1, a_3\}\}$.

for any query operator \oplus , we define an *extended operator* $\hat{\oplus}$ with an analogous meaning that operates on *CPDB*. It is defined by $\hat{\oplus} = (\oplus, \tau_{\oplus})$ where τ_{\oplus} is a function that produces the descriptive sentence of a result based on the descriptive sentences of the operands in a manner that is appropriate for operation \oplus . Obviously, a thusly expressed query \hat{Q} on a compact probabilistic database *CPDB* should adhere to the semantics above and Equation 4:

$$\hat{Q}(CPDB) = \bigcup_{w \in W(CPDB)} Q(w) = \bigcup_{\bar{\varphi} \in \Phi(\Omega)} \{a \mid \langle a, \varphi \rangle \in \hat{Q}(\widehat{DB}) \wedge \bar{\varphi} \Rightarrow \varphi\} \quad (5)$$

2.2 Definition of JudgeD, a probabilistic datalog

As a representation formalism in which both probabilistic data as well as soft and hard rules can be expressed, we choose JudgeD, a probabilistic datalog [17]. Several probabilistic logics have been proposed in the last decades among others pD [18] and ProbLog [10]. In these logics probabilities can be attached to facts and rules. JudgeD is obtained by defining in the abovedescribed formalism that a data item is a fact or rule. Moreover, datalog entailment is extended with sentence manipulation [16]. The thus obtained probabilistic datalog is as expressive as ProbLog regarding dependency relationships.

Probabilistic datalog. We base our definition of Datalog on [19, Ch.6] (only positive Datalog for simplicity). We postulate disjoint sets *Const*, *Var*, *Pred* as the sets of *constants*, *variables*, and *predicate symbols*, respectively. Let $c \in \text{Const}$, $X \in \text{Var}$, and $p \in \text{Pred}$. A *term* $t \in \text{Term}$ is either a constant or variable where $\text{Term} = \text{Const} \cup \text{Var}$. An *atom* $A = p(t_1, \dots, t_n)$ consists of an n -ary predicate symbol p and a list of argument terms t_i . An atom is *ground* iff $\forall i \in 1..n : t_i \in \text{Const}$. A *clause* or *rule* $r = (A^h \leftarrow A_1, \dots, A_m)$ is a Horn clause representing the knowledge that A^h is true iff all A_i are true. A *fact* is a rule without body ($A^h \leftarrow$). A set KB of rules is called a *knowledge base* or *program*. The usual safety conditions of pure Datalog apply.

Let $\theta = \{X_1/t_1, \dots, X_n/t_n\}$ be a *substitution* where X_i/t_i is called a *binding*. $A\theta$ and $r\theta$ denote the atom or rule obtained by replacing (as defined by θ) each X_i occurring in A or r respectively by the corresponding term t_i .

Let $(A^h \stackrel{\varphi}{\leftarrow} A_1, \dots, A_m)$ denote the tuple $\langle A^h \leftarrow A_1, \dots, A_m, \varphi \rangle$. Note that this not only allows for the specification of uncertain facts, but also uncertain rules as well as dependencies between the existence of facts and rules using the sentences φ .

Probabilistic entailment. Entailment is defined as follows:

$$\begin{array}{l} r \in \text{KB} \quad r = (A^h \stackrel{\varphi}{\leftarrow} A_1, \dots, A_m) \\ \exists \theta : A^h\theta \text{ is ground} \wedge \forall i \in 1..m : \text{KB} \models \langle A_i\theta, \varphi_i \rangle \\ \varphi' = \varphi \wedge \bigwedge_{i \in 1..m} \varphi_i \quad \varphi' \neq \perp \\ \hline \text{KB} \models \langle A^h\theta, \varphi' \rangle \end{array}$$

In other words, given a rule r from the knowledge base and a substitution θ that makes the atoms A_i in the body true for sentences φ_i , then we can infer the

```

a1  annot(id-ph,pos1-2,hotel) [x=1].      @p(x=1) = 0.5.
a2  annot(id-ph,pos1-2,person) [x=2].     @p(x=2) = 0.4.
a3  annot(id-ph,pos1-2,fragrance) [x=3].  @p(x=3) = 0.1.
a4  annot(id-p,pos1,firstname) [y=1].     @p(y=1) = 0.3.
a5  annot(id-p,pos1,city) [y=2].         @p(y=2) = 0.7.

a6  contained(pos1,pos1-2).
a7  hardrule :- annot(Ph1,P1,city), annot(Ph2,P2,person), contained(P1,P2).

```

Fig. 3. Paris Hilton example (simplified) in JudgeD (sentences in square brackets; ‘@p’ syntax specifies probabilities).

substituted atom $A^h\theta$ with a sentence that is a conjunction of all φ_i and the sentence φ of the rule r (unless this conjunction is inconsistent). This definition of probabilistic entailment is obtained from applying the querying framework of Section 2.1 to normal datalog entailment [16]. It can be proven to be consistent with Equation (5).

2.3 Representing PDI Results in JudgeD

Probabilistic data integration (PDI) is a specific kind of data integration where extraction and integration problems are handled by means of a probabilistic data representation. In this section, we illustrate JudgeD by showing how to represent an information extraction result.

In the Paris Hilton example, the initial information extraction produces uncertain annotations: the type of the phrase “Paris Hilton” occurring as the first and second word of the sentence can be either a hotel, person, or fragrance with, for example, probabilities 0.5, 0.4, 0.1, respectively. Furthermore, the first word “Paris” can either be a firstname or a city with, for example, probabilities 0.3 and 0.7, respectively.. We can represent this in JudgeD as in Figure 3 (a_1 – a_5). Probabilities are obtained from classifiers or scoring or ranking functions used in information extraction and data integration.

A user may want to express evidence that words contained in phrases interpreted as persons, should not be interpreted as cities. If we absolutely trust this to be true, we express this as a *hard rule*. In contrast, a *soft rule* is a rule that is only partially trusted, i.e., the evidence is uncertain. In JudgeD we can express the evidence by rule *hardrule* in Figure 3 (a_6 – a_7). Executing this rule provides the information under which conditions the rule is true, in this case, $x=2 \wedge y=2$. In this case, it is a negative rule, i.e., we ‘observe’ the evidence that *hardrule* is false. As we will see in the next section, this evidence can be incorporated by conditioning and rewriting the database on $\neg(x=2 \wedge y=2)$.

3 Conditioning

As the example in Section 2.3 illustrates, our approach is to specify evidence by rules. Since a rule may only be true in a subset of worlds, the rule actually specifies which worlds are consistent with the evidence. By executing the rule,

we obtain this information in terms of the *evidence sentence* φ_e . To incorporate such evidence means that the database³ needs to be *conditioned*.

A common way of conditioning in probabilistic programming [10, 11] is to extend inference with an **observe** capability. Here, we propose to *rewrite* the database into an equivalent one that no longer contains **observe** statements: the evidence is directly *incorporated* in the probabilistic data. By ensuring that evidence incorporation can be done iteratively, the “Improve data quality” step of Figure 1 can be realized without an ever-growing set of **observe** statements.

The intuition of conditioning is to eliminate all worlds that are inconsistent with the evidence and redistribute the eliminated probability mass over the remaining worlds by means of normalization. This can be realized directly on the compact probabilistic database by constructing an adjusted set of partitionings Ω' , rewriting the sentences of the data items, and removing any data items for which the sentence becomes inconsistent (i.e., \perp).

The approach is presented in several steps: Section 3.1 defines the semantics of a probabilistic database with evidence. Section 3.2 explains how to reduce conditioning with a complex set of evidences to one or more simple conditionings. Section 3.3 explains how to rewrite the original database into a conditioned one whereby we focus on hard rules first. Section 3.4 explains how to condition with soft rules. We conclude this section with a discussion on iterative conditioning.

3.1 Semantics of a database with evidence

We abstractly denote evidence as a set E of queries/rules that should be true (positive evidence). We extend the definition of $CPDB = \langle \widehat{DB}, \Omega, P \rangle$ to a *compact probabilistic database with evidence* $CPDBE = \langle \widehat{DB}, \Omega, P, E \rangle$ with semantics

$$W(CPDBE) = \{w \mid w \in W(CPDB) \wedge \forall Q_e \in E : Q_e(w) \text{ is true}\}$$

Concrete probabilistic database formalisms may provide specific mechanisms for specifying evidence. For JudgeD, we extend the language with a specific kind of rule: **observe**(A_e). A program containing k observed atoms A_e^i ($i \in 1..k$) defines $E = \{A_e^1, \dots, A_e^k\}$.

An evidence query $Q_e^i \in E$ has exactly two results: $Q_e^i(CPDB) = \{\langle true, \varphi_i \rangle, \langle false, \neg\varphi_i \rangle\}$. Since evidence filters worlds that are inconsistent with it, we determine an *evidence sentence* $\varphi_e = \bigwedge_{i \in 1..k} \varphi_i$. We use E and φ_e interchangeably:

$$W(CPDBE) = \{w \mid w \in W(CPDB) \wedge \varphi_e\} \tag{6}$$

The probability mass associated with eliminated worlds is redistributed over the remaining worlds by means of normalization.

$$P_e(\varphi) = \frac{P(\varphi \wedge \varphi_e)}{P(\varphi_e)} \tag{7}$$

³ Note that we also refer to a JudgeD program as a database.

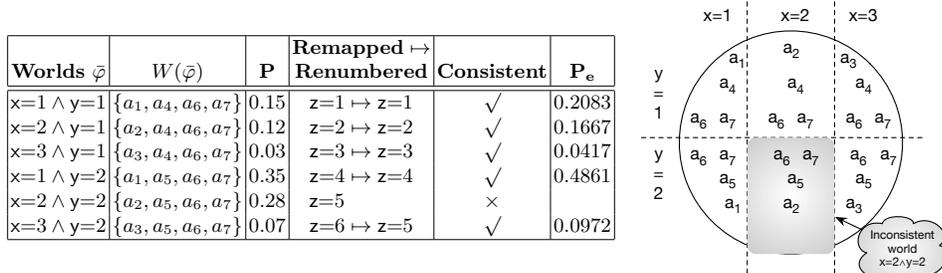


Fig. 4. Illustration of partitioning remapping

Querying is extended in a straightforward manner by adapting Equation (5):

$$\begin{aligned}
\hat{Q}(CPDBE) &= \bigcup_{w \in W(CPDBE)} Q(w) \\
&= \bigcup_{\bar{\varphi} \in \Phi(\Omega), \bar{\varphi} \Rightarrow \varphi_e} \{a \mid \langle a, \varphi \rangle \in \hat{Q}(\widehat{DB}) \wedge \bar{\varphi} \Rightarrow \varphi\}
\end{aligned} \tag{8}$$

3.2 Remapping partitionings

Figure 4 illustrates that in the Paris Hilton example of Figure 3, partitions x^3 and y^2 that were independent now become dependent because one of the six possible worlds is inconsistent with the evidence $\varphi_e = \neg(x=2 \wedge y=2)$. When this happens, we *remap* x and y , i.e., replace them with a fresh partitioning z^6 representing their combined possibilities. By simple logical equivalence, we can find formulas for the labels of the original partitionings, for example, $x=1 \Leftrightarrow (z=1 \vee z=4)$. These can be used to rewrite sentences based on x and y to sentences based on z . Since worlds and their contents are determined by sentences and these sentences are replaced by equivalent ones, this remapping of two or more partitionings to a single fresh one is idempotent.

Remapping. For a sentence containing more than one partitioning, the partitionings may become dependent and remapping is necessary. Let $\Omega_e = \omega(\varphi_e) = \{\omega^{n_1}, \dots, \omega^{n_k}\}$ be the set of partitionings to be remapped. We introduce a fresh partitioning $\bar{\omega}^n$ where $n = n_1 \times \dots \times n_k$. Let the bijection $\lambda_{\Omega_e} : \Phi(\Omega_e) \leftrightarrow L(\bar{\omega}^n)$ be the *remapping function*. A valid remapping function can be constructed in a straightforward way by viewing the values in the labels of the partitionings of a full sentence as a vector of numbers v_1, \dots, v_k and computing the value v in the label of $\bar{\omega}^n$ as $v = 1 + \sum_{i \in 1..k} (v_i - 1) \prod_{j \in i+1..k} n_j$. For example, $\lambda_{\Omega_e}(x=3 \wedge y=2) = (z=6)$ because $1 + (3 - 1) \times 2 + (2 - 1) \times 1 = 6$.

A sentence φ can be rewritten into $\lambda_{\Omega_e}(\varphi)$ by replacing every label $l_{ij} = (\omega_i = v_i^j)$ with $\bigvee_{l \in L(\bar{\omega}^n), l_{ij} \in \lambda_{\Omega_e}^{-1}(l)} l$. For example, $\lambda_{\Omega_e}(x=1 \wedge y=2) = ((z=1 \vee z=4) \wedge (z=4 \vee z=5 \vee z=6)) = (z=4)$. Observe that, since all partitionings in a sentence are rewritten into a single one, the rewritten evidence sentence is of the form $\lambda_{\Omega_e}(\varphi_e) = (\bar{\omega}^n = v_1) \vee \dots \vee (\bar{\omega}^n = v_m)$ for some m .

Finally, given φ_e , a compact probabilistic database $CPDB = \langle \widehat{DB}, \Omega, P \rangle$ can be rewritten into $\lambda_{\Omega_e}(CPDB) = \langle \widehat{DB}', \Omega', P' \rangle$ where

$$\widehat{DB}' = \{ \langle a, \lambda_{\Omega_e}(\varphi) \rangle \mid \langle a, \varphi \rangle \in \widehat{DB} \} \quad (9)$$

$$\Omega' = (\Omega \setminus \Omega_e) \cup \{ \bar{\omega}^n \} \quad (10)$$

$$P'(l) = \begin{cases} P(\lambda_{\Omega_e}^{-1}(l)) & \text{if } l \in L(\bar{\omega}^n) \\ P(l) & \text{otherwise} \end{cases}$$

Splitting. If many partitionings are involved, remapping may introduce partitionings ω^n with large n . Note, however, that the procedure is only necessary if the partitionings become independent due to the evidence. For example, if the evidence would be $\varphi_e = \neg(x=3) \wedge y=2$, x and y remain independent. Therefore, we first *split* φ_e into independent components and treat them separately.

First φ_e is brought into conjunctive normal form $\varphi_1 \wedge \dots \wedge \varphi_n$ whose conjuncts are then ‘clustered’ into m independent *components* $\varphi_e^i = \varphi_{j_1} \wedge \dots \wedge \varphi_{j_k}$ ($i \in 1..m$) such that for maximal m , every conjunct is in exactly one component, and for every pair of components φ_e^1 and φ_e^2 , it holds $\omega(\varphi_e^1) \cap \omega(\varphi_e^2) = \emptyset$.

Note that, because of independence between partitionings, the components specify independent evidence that can be incorporated separately. In the sequel, we denote with φ_e a single component of the evidence sentence. Furthermore, since remapping reduces an evidence sentence to one based on one partitioning, splitting and remapping together simplify conditioning to one or more conditionings on single partitionings.

3.3 Conditioning with hard rules by means of program rewriting

Given $CPDBE = \langle \widehat{DB}, \Omega, P, \varphi_e \rangle$, let $CPDB = \langle \widehat{DB}'', \Omega'', P'' \rangle = A_{\varphi_e}(CPDBE)$ be a rewritten compact probabilistic database that incorporates φ_e in the probabilistic data itself. We define $A_{\varphi_e}(CPDBE)$ as follows. The partitionings $\Omega_e = \omega(\varphi_e)$ are remapped to fresh partitioning $\bar{\omega}$ using remapping function λ_{Ω_e} . Effectuating this remapping obtains $\langle \widehat{DB}', \Omega', P' \rangle = \lambda_{\Omega_e}(\langle \widehat{DB}, \Omega, P \rangle)$. The component φ_e itself can also be rewritten into $\bar{\varphi}_e = \lambda_{\Omega_e}(\varphi_e)$ which results in a sentence of the form $\bar{\varphi}_e = \bar{l}_1 \vee \dots \vee \bar{l}_m$ where $\bar{l}_j = (\bar{\omega} = v_j)$ for some m .

The evidence sentence $\bar{\varphi}_e$ specifies which worlds $W(\langle \widehat{DB}', \Omega', P' \rangle)$ are valid, namely those identified by each \bar{l}_j . Let $L = \{ \bar{l}_1, \dots, \bar{l}_m \}$. The worlds identified by $\bar{L} = L(\bar{\omega}) \setminus L$ are inconsistent with $\bar{\varphi}_e$. This can be effectuated in \widehat{DB}' by setting labels identifying inconsistent worlds to \perp in all sentences occurring in \widehat{DB}' . A descriptive assertion for which the sentence becomes \perp can be deleted from the database as it is no longer present in any remaining world.

Let $\lambda_{\bar{L}}(\varphi)$ be the sentence obtained by setting l to \perp in φ for each $l \in \bar{L}$. We can now define \widehat{DB}'' as follows:

$$\widehat{DB}'' = \{ \langle a, \lambda_{\bar{L}}(\varphi) \rangle \mid \langle a, \varphi \rangle \in \widehat{DB}' \wedge \lambda_{\bar{L}}(\varphi) \neq \perp \}$$

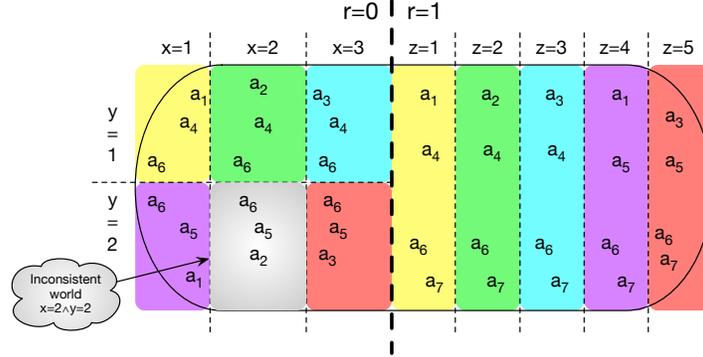


Fig. 5. Illustration of applying a soft rule.

Finally, the probability mass of the inconsistent worlds needs to be redistributed over the remaining consistent ones. Furthermore, since some labels \bar{l}_j representing these inconsistent worlds should obtain a probability $P''(\bar{l}_j) = 0$, these labels should be removed, and because we assume the values of a partitioning ω^n to range from 1 to n , we renumber them by replacing $\bar{\omega}^n$ with $\hat{\omega}^m$.

Let $\Omega'' = (\Omega' \setminus \{\bar{\omega}^n\}) \cup \{\hat{\omega}^m\}$. The bijection $f : L(\hat{\omega}^m) \leftrightarrow L$ uniquely associates each new ‘renumbered’ label with an original label of a consistent world. In \widehat{DB}'' replace every occurrence of a label $\bar{l}_j \in L$ with $f(\bar{l}_j)$. Note that labels from \bar{L} will no longer occur in \widehat{DB}'' . P'' is defined by setting the probabilities of the new labels as follows: $P''(\bar{l}_j) = \frac{1}{p}P'(f(\bar{l}_j))$ where $p = \sum_{\bar{l}_j \in L} P'(\bar{l}_j)$.

In the next section, we turn **hardrule** into a soft rule and show what the end result for the conditioned Paris Hilton example looks like (see Figure 6).

3.4 Conditioning with soft rules

A soft rule is an uncertain hard rule, hence the same principle of probabilistic data can be used to represent a soft rule: with a partitioning ω_r^2 where labels $\omega_r^2=0$ and $\omega_r^2=1$ identify all worlds where the rule is false and true, respectively. For Figure 3, we write

```
a7 softrule :- annot(Ph1,P1,city), annot(Ph2,P2,person), contained(P1,P2) [r=1].
```

which effectively means that $\langle a_7, \top \rangle$ is replaced with $\langle a_7, r=1 \rangle$ in the database. We now have 12 worlds in Figure 4: the original 6 ones, and those 6 again but without a_7 .

Executing **softrule** results in $\{\langle true, x=2 \wedge y=2 \wedge r=1 \rangle, \langle false, \neg(x=2 \wedge y=2 \wedge r=1) \rangle\}$. Since it is a negative rule, $\varphi_e = \neg(x=2 \wedge y=2 \wedge r=1)$. Instead of direct conditioning for this evidence, we strive for the possible worlds as illustrated in Figure 5. Depicted here are the original worlds in case $r=0$ and the conditioned situation in case $r=1$. It can be obtained by conditioning the database as if it was a hard rule, but effectuate the result only for worlds for which $r=1$.

a_1	<code>annot(id-ph,pos1-2,hotel)</code> [[r=0 and x=1) or (r=1 and (z=1 or z=4))].	$\text{p}(x=1) = 0.5.$ $\text{p}(x=2) = 0.4.$
a_2	<code>annot(id-ph,pos1-2,person)</code> [[r=0 and x=2) or (r=1 and z=2)].	$\text{p}(x=3) = 0.1.$ $\text{p}(y=1) = 0.3.$
a_3	<code>annot(id-ph,pos1-2,fragrance)</code> [[r=0 and x=3) or (r=1 and (z=3 or z=5))].	$\text{p}(y=2) = 0.7.$ $\text{p}(z=1) = 0.2083.$
a_4	<code>annot(id-p,pos1,firstname)</code> [[r=0 and y=1) or (r=1 and (z=1 or z=2 or z=3))].	$\text{p}(z=2) = 0.1667.$ $\text{p}(z=3) = 0.0417.$
a_5	<code>annot(id-p,pos1,city)</code> [[r=0 and y=2) or (r=1 and (z=4 or z=5))].	$\text{p}(z=4) = 0.4861.$ $\text{p}(z=5) = 0.0972.$
a_6	<code>contained(pos1,pos1-2).</code>	$\text{p}(r=1) = 0.8.$ $\text{p}(r=2) = 0.2.$
a_7	<code>softrule :- annot(Ph1,P1,city), annot(Ph2,P2,person), contained(P1,P2) [r=1].</code>	

Fig. 6. Paris Hilton example with evidence of `softrule` incorporated as a soft rule.

Soft rule rewriting. Given $CPDBE = \langle \widehat{DB}, \Omega, P, \varphi_e \rangle$ and φ_e is a soft rule governed by partitioning ω_r . Let \widehat{DB}' and φ_e' be the counterparts of \widehat{DB} and φ_e where in all sentences $\omega_r=1$ is set to \top and $\omega_r=0$ to \perp . Let $\Omega' = \Omega \setminus \{\omega_r\}$. Let P' be P restricted to the domain of Ω' . This effectively makes the rule a hard rule. Let $\langle \widehat{DB}'', \Omega'', P'' \rangle = A_{\varphi_e}(\langle \widehat{DB}', \Omega', P', \varphi_e' \rangle)$ be the database that incorporates the evidence as a hard rule.

From this result we construct a probabilistic database that contains both the data items from the original worlds when $\omega_r=0$ and the data items from the rewritten worlds when $\omega_r=1$. We define $A_{\varphi_e}(CPDBE) = \langle \widehat{DB}''', \Omega''', P''' \rangle$ where

$$\begin{aligned} \widehat{DB}''' &= \{ \langle a, (\varphi_1 \wedge \omega_r=0) \vee (\varphi_2 \wedge \omega_r=1) \rangle \mid \langle a, \varphi_1 \rangle \in \widehat{DB} \wedge (\omega_r=0 \Rightarrow \varphi_1) \wedge \langle a, \varphi_2 \rangle \in \widehat{DB}'' \} \\ &\quad \cup \{ \langle a, (\varphi_1 \wedge \omega_r=0) \rangle \mid \langle a, \varphi_1 \rangle \in \widehat{DB} \wedge (\omega_r=0 \Rightarrow \varphi_1) \wedge \langle a, \varphi_2 \rangle \notin \widehat{DB}'' \} \\ &\quad \cup \{ \langle a, (\varphi_2 \wedge \omega_r=1) \rangle \mid \langle a, \varphi_1 \rangle \in \widehat{DB} \wedge (\omega_r=0 \not\Rightarrow \varphi_1) \wedge \langle a, \varphi_2 \rangle \in \widehat{DB}'' \} \\ \Omega''' &= \Omega \cup \Omega'' \\ P''' &= P \cup P'' \end{aligned}$$

See Figure 6 for the conditioned database of the Paris Hilton example.

3.5 Iterative conditioning

The intention is to use this approach iteratively, i.e., whenever new evidence is specified, the evidence is directly incorporated. One may wonder what happens if the same rule is incorporated twice.

With hard rules the answer is simple: since all worlds inconsistent with the rule have been filtered out, all remaining rules are consistent with the rule, i.e., when the evidence is a rule that has already been incorporated $\varphi_e = \top$.

In case of soft rules, all original worlds, hence also the ones inconsistent with the rule, are still present (see Figure 5). Observe, however, that all inconsistent worlds have $r=0$ in their full sentences. Applying the rule again, will leave all original worlds unaffected, because in those worlds the rule is not present. And where the rule is true, the worlds inconsistent with the rule have already been filtered out. Therefore, also for soft rules it holds that re-incorporating them leaves the database unaffected.

If, however, a soft rule $\langle \mathbf{r}, r1=1 \rangle$ is incorporated again but governed by a different partitioning, i.e., $\langle \mathbf{r}, r2=1 \rangle$, different probabilities for query answers are obtained. Note, however, that this pertains to a different situation: with both evidences based on $r=1$, the evidence effectively comes from the same source twice, which provides no new evidence and the result is the same. With evidences based on different partitions, the evidence effectively comes from two different sources. Indeed, this provides extra independent evidence, hence probabilities are conditioned twice.

Scalability. There are two main classes of probabilistic databases: relational PDBs and probabilistic logics. The first step of evaluating the evidence rule to obtain the evidence sentence φ_e has the same complexity as querying in such systems. Remapping and redistribution of probabilities depends on $|\Omega_e|$ which is exponential in the number of partitionings involved in φ_e . We assume that uncertainty remains fairly local, i.e., after splitting only components with few partitionings remain. The same holds for simplification and normal form reduction of the sentences. Database rewriting affects all data items referring to remapped partitionings, which is worst case linear in the size of the database. The result is a probabilistic database again with at most the same size but possibly longer sentences, i.e., the complexity of querying the resulting database does not change. In short, assuming uncertainty remains local, algorithms implementing our approach are expected to be well-scalable.

4 Validation

The main proof obligation is that the database without evidence obtained by $A_{\varphi_e}(CPDBE)$ represents the same possible worlds as the original $CPDBE$.

Theorem 1. $W(A_{\varphi_e}(CPDBE)) = W(CPDBE)$

Proof sketch. The proof sketch is based on showing that in each of the steps, the possible worlds remain the same. The first step splits the evidence sentence into independent components. Let $\varphi_e = \varphi_1 \wedge \varphi_2$. Since $W(CPDBE) = \{w \mid w \in W(CPDB) \wedge \varphi_e\}$ (see Equation 6) and φ_1 and φ_2 share no partitionings, the filtering of worlds on $\varphi_1 \wedge \varphi_2$ is the same as filtering first on φ_1 and then on φ_2 .

The second step is the remapping of the partitionings in the evidence sentence component. The remapping introduces a single fresh partitioning $\bar{\omega}^n$. Note that the remapping function λ_{Ω_e} is a bijection uniquely relating each full sentence $\bar{\varphi}$ constructed from $\Phi(\Omega_e)$ with one label $\bar{l} \in L(\bar{\omega}^n)$. In other words, $W(\bar{\varphi}) = W(\bar{l})$ hence the possible worlds remain the same (see Equations 2, 4, and 9)

$$\begin{aligned} W(CPDB) &= \{DB \mid \bar{\varphi} \in \Phi(\Omega) \wedge DB = \{a \mid \langle a, \varphi \rangle \in \widehat{DB} \wedge \bar{\varphi} \Rightarrow \varphi\}\} \\ &= \{DB \mid \bar{l} \in L(\bar{\omega}^n) \wedge DB = \{a \mid \langle a, \lambda_{\Omega_e}(\varphi) \rangle \in \widehat{DB} \wedge \bar{l} \Rightarrow \lambda_{\Omega_e}(\varphi)\}\} \end{aligned}$$

Since $\lambda_{\Omega_e}(\varphi)$ replaces every label with an equivalent disjunction of fresh labels $\bar{\varphi} \Rightarrow \varphi$ is true whenever $\bar{l} \Rightarrow \lambda_{\Omega_e}(\varphi)$ is true. Therefore, remapping retains the same possible worlds. This can also be illustrated with Figure 4. The six possible worlds in a 2-by-3 grid are remapped to a 1-by-6 grid containing the same distribution of assertions.

The above steps have transformed $W(CPDBE)$ into

$$W(CPDBE) = \{DB \mid \bar{l} \in L(\bar{\omega}^n) \\ \wedge DB = \{a \mid \langle a, \lambda_{\Omega_e}(\varphi) \rangle \in \widehat{DB} \wedge \bar{l} \Rightarrow \lambda_{\Omega_e}(\varphi)\} \\ \wedge \lambda_{\Omega_e}(\varphi_e)\}$$

It has already been noticed that, $\lambda_{\Omega_e}(\varphi_e)$ is of the form $\lambda_{\Omega_e}(\varphi_e) = (\bar{\omega}^n=v_1) \vee \dots \vee (\bar{\omega}^n=v_m)$ for some m . The third step is setting labels identifying inconsistent worlds to \perp , i.e., labels $\bar{l} \notin \{(\bar{\omega}^n=v_1), \dots, (\bar{\omega}^n=v_m)\}$. Figure 4 illustrates how the world identified by $z=5$ is eliminated, and the resulting database is

$$\{\langle a_1, z=1 \vee z=4 \rangle, \langle a_2, z=2 \rangle, \langle a_3, z=3 \vee z=6 \rangle, \langle a_4, z=1 \vee z=2 \vee z=3 \rangle, \langle a_5, z=4 \vee z=6 \rangle, \\ \langle a_6, z=1 \vee z=2 \vee z=3 \vee z=4 \vee z=6 \rangle, \langle a_7, z=1 \vee z=2 \vee z=3 \vee z=4 \vee z=6 \rangle\}$$

The label renumbering for $\bar{\omega}^n$ and redistribution of probability mass to labels $(\bar{\omega}^n=v_1), \dots, (\bar{\omega}^n=v_m)$ in the remapped label space is equivalent with Equation 7.

Figure 4 illustrates how the worlds remaining in $W(CPDBE) = \{w \mid w \in W(CPDB) \wedge \varphi_e\}$ (Equation 6) after applying a soft rule are constructed by effectively taking the union of the $\omega_r=0$ *partition* of $W(CPDB)$ with the rewritten worlds of the $\omega_r=1$ partition of $W(CPDB)$.

5 Conclusions

The main contribution of this paper is an iterative approach for incorporating evidence of users in probabilistically integrated data, evidence which can be specified both as hard and soft rules. This capability makes the two-phase probabilistic data integration process possible where in the second phase, the use of integrated data could lead to evidence which can continuously improve the data quality. The benefit is that a data integration result can be more quickly obtained as it can be imperfect.

The first objective for future work is the engineering aspect of the approach: developing a software prototype with the purpose of investigating the scalability of the approach. Furthermore, more future work is needed to complete and improve aspects of the PDI process such as indeterministic approaches for other data integration problems, improving the scalability of probabilistic database technology, and application of PDI to real-world scenarios and data sizes.

References

1. van Keulen, M.: Probabilistic data integration. In Sakr, S., Zomaya, A., eds.: Encyclopedia of Big Data Technologies. Springer (2018) 1–9

2. van Keulen, M., de Keijzer, A.: Qualitative effects of knowledge rules and user feedback in probabilistic data integration. *VLDB Journal* **18**(5) (2009) 1191–1217
3. van Keulen, M.: Managing uncertainty: The road towards better data interoperability. *IT - Information Technology* **54**(3) (2012) 138–146
4. Magnani, M., Montesi, D.: A survey on uncertainty management in data integration. *JDIQ* **2**(1) (2010) 5:1–5:33
5. Dalvi, N., Ré, C., Suciu, D.: Probabilistic databases: Diamonds in the dirt. *Communications of the ACM* **52**(7) (2009) 86–94
6. Suciu, D., Olteanu, D., Ré, C., Koch, C.: Probabilistic databases. *Synthesis Lectures on Data Management* **3**(2) (2011) 1–180
7. Panse, F., van Keulen, M., Ritter, N.: Indeterministic handling of uncertain decisions in deduplication. *JDIQ* **4**(2) (2013) 9:1–9:25
8. Wanders, B., van Keulen, M., van der Vet, P.: Uncertain groupings: Probabilistic combination of grouping data. In: *Proc. of Int’l Conf. on Database and Expert Systems Applications (DEXA)*. Volume 9261 of LNCS., Springer (2015) 236–250
9. Habib, M., Van Keulen, M.: TwitterNEED: A hybrid approach for named entity extraction and disambiguation for tweet. *Natural Language Engineering* **22** (2016) 423–456
10. Raedt, L.D., Kimmig, A., Toivonen, H.: ProbLog: a probabilistic prolog and its application in link discovery. In: *Int’l Joint Conf. on Artificial Intelligence (IJCAI)*, AAAI Press (2007) 2468–2473
11. Olmedo, F., Gretz, F., Jansen, N., Kaminski, B.L., Katoen, J.P., Mciver, A.: Conditioning in probabilistic programming. *ACM Trans. Program. Lang. Syst.* **40**(1) (2018) 4:1–4:50
12. Theobald, M., De Raedt, L., Dylla, M., Kimmig, A., Miliaraki, I.: 10 years of probabilistic querying – what next? In: *Proc. of East European Conf. on Advances in Databases and Information Systems (ADBIS)*. Volume 8133 of LNCS., Springer (2013) 1–13
13. Koch, C., Olteanu, D.: Conditioning probabilistic databases. *Proc. VLDB Endowment* **1**(1) (2008) 313–325
14. van Keulen, M., Habib, M.: Handling uncertainty in information extraction. In: *Proc. of Int’l Conf. on Uncertainty Reasoning for the Semantic Web (URSW)*. Volume 778 of CEUR-WS. (2011) 109–112
15. Jayram, T.S., Krishnamurthy, R., Raghavan, S., Vaithyanathan, S., Zhu, H.: Avatar information extraction system. *IEEE Data Eng. Bull.* **29**(1) (2006) 40–48
16. Wanders, B., van Keulen, M.: Revisiting the formal foundation of probabilistic databases. In: *Conf. of the Int’l Fuzzy Systems Association and the European Society for Fuzzy Logic and Technology, IFSA-EUSFLAT*, Atlantis Press (2015) 47
17. Wanders, B., van Keulen, M., Flokstra, J.: JudgeD: a probabilistic datalog with dependencies. In: *Proc. of Workshop on Declarative Learning Based Programming, DeLBP*. Number WS-16-07, AAAI (2016)
18. Fuhr, N.: Probabilistic datalog: a logic for powerful retrieval methods. In: *Int’l Conf. on Research and Development in Information Retrieval (SIGIR)*, ACM (1995) 282–290
19. Ceri, S., Gottlob, G., Tanca, L.: *Logic Programming and Databases*. Springer (1990) ISBN 3-540-51728-6.