16th World Congress of the International Fuzzy Systems Association (IFSA)
9th Conference of the European Society for Fuzzy Logic and Technology (EUSFLAT)

Revisiting the formal foundation of Probabilistic Databases

Brend Wanders[1] **Maurice van Keulen**[1]

[1] Faculty of EEMCS, University of Twente, Enschede, The Netherlands. {b.wanders,m.vankeulen}@utwente.nl

Abstract

One of the core problems in soft computing is dealing with uncertainty in data. In this paper, we revisit the formal foundation of a class of probabilistic databases with the purpose to (1) obtain data model independence, (2) separate metadata on uncertainty and probabilities from the raw data, (3) better understand aggregation, and (4) create more opportunities for optimization. The paper presents the formal framework and validates data model independence by showing how to a obtain probabilistic Datalog as well as a probabilistic relational algebra by applying the framework to their non-probabilistic counterparts. We conclude with a discussion on the latter three goals.

Keywords: probabilistic databases, probabilistic Datalog, probabilistic relational algebra, formal foundation

1. Introduction

One of the core problems in soft computing is dealing with uncertainty in data. For example, many data activities such as data cleaning, coupling, fusion, mapping, transformation, information extraction, etc. are about dealing with the problem of semantic uncertainty [1, 2]. In the last decade, there has been much attention in the database community to scalable manipulation of uncertain data. Probabilistic database research produced numerous uncertainty models and research prototypes, mostly relational; see [3, Chp.3] for an extensive survey.

In our research we actively apply this technology for soft computing data processing tasks such as indeterministic deduplication [4], probabilistic XML data integration [5], and probabilistic integration of data about groupings [6]. Based on these experiences, we find that there are still important open problems in dealing with uncertain data and that the available systems are inadequate on certain aspects. We address the following four aspects.

Data model dependence Depending on the requirements and domain, we use different data models such as relational, XML, and RDF. The available models for uncertain data are tightly connected to a particular data model resulting in a non-uniform dealing with uncertain data as well as replication of functionality in the various prototype systems.

Insufficient understanding of core concepts Uncertainty in data has been the subject of research in several research communities for decades. Nevertheless, we believe our understanding of certain concepts is not deep enough. For example, *truth* of facts that are uncertain. Or, what are *possible worlds* really? Also, many models support possible alternatives in some way often associated with a probability. Are these *probabilities* truly add-ons or are they tightly connected to the alternatives?

Aggregates In many data processing tasks, being able to aggregate data in multiple ways is essential. Computing aggregates over uncertain data is, however, inherently exponential. There is much work on approximating aggregates, often with error bounds, but this does not seem to suffice in all cases. Furthermore, systems offer operations on uncertain data as aggregates, such as EXP (expected value) in Trio [7]; they seem different from traditional aggregates such as SUM, or is there a more generic concept of aggregation that encompasses all?

Optimization opportunities There has been some work on optimization for probabilistic databases, for example, in the context of MayBMS/SPROUT [8, 9], but as we experienced in [6], where we apply MayBMS to a bio-informatics homology use case, the research prototypes do not scale well enough to thousands of random variables. By generalizing certain concepts in our formal foundation, we hope to create better understanding of optimization opportunities.

Contributions We address the above with a new formalisation of a probabilistic database and associated notions as a result of revisiting its fundaments. The formalization has the following properties:

- Data model independent
- Meta-data about uncertain data loosely coupled to raw data
- Loosely coupled probabilities
- Unified view on *aggregates* and *probabilistic database-specific functions*

We demonstrate the usefulness of the formalization for creating more insight by discussing questions like "What *are* possible worlds?", "What is truth in an uncertain context?", "What are aggregates?", and "What optimization opportunities come to light?". Furthermore, we validate the data model independence by illustrating how a probabilistic Datalog

"Paris Hilton stayed in the Paris Hilton"

	phrase	pos	refers to
1	Paris Hilton	1,2	the person
2	Paris Hilton	1,2	the hotel
3	Paris	1	the capital of France
4	Paris	1	Paris, Ontario, Canada
5	Hilton	2	the hotel chain
6	Paris Hilton	6,7	the person
7	Paris Hilton	6,7	the hotel
8	Paris	6	the capital of France
9	Paris	6	Paris, Ontario, Canada
10	Hilton	7	the hotel chain
	⋮	⋮	⋮

Figure 1: Example natural language sentence with a few candidate annotations [10].

as well as a probabilistic relational database can be defined using our framework.

Running example We use natural language processing as a running example, the sub-task of Named Entity Extraction and Disambiguation (NEED) in particular. NEED attempts to detect *named entities*, i.e., phrases that refer to real-world objects. Natural language is ambiguous, hence the NEED process is inherently uncertain. The example sentence of Figure 1 illustrates this: "Paris Hilton" may refer to a person (the American socialite, television personality, model, actress, and singer) or to the hotel in France. In the latter case, the sub-phrase "Paris" refers to the capital of France although there are many more places and other entities with the name "Paris" (e.g., see http://en.wikipedia.org/wiki/Paris_(disambiguation) or a gazetteer like GeoNames[1]).

A human immediately understands all this, but to a computer this is quite elusive. One typically distinguishes different kinds of ambiguity such as [11]: (a) semantic ambiguity (to what class does an entity phrase belong, e.g., does "Paris" refer to a name or a location?), (b) structural ambiguity (does a word belong to the entity or not, e.g., "Lake Garda" vs. "Garda"?), and (c) reference ambiguity (to which real world entity does a phrase refer, e.g., does "Paris" refer to the capital of France or one of the other 158 Paris instances found in GeoNames?). We represent detected entities and the uncertainty surrounding them as *annotation candidates*. Figure 1 contains a table with a few for the example sentence.

NEED typically is a multi-stage process where voluminous intermediary results need to be stored and manipulated. Furthermore, the dependencies between the candidates should be carefully maintained. For example, "Paris Hilton" can be a person or hotel, but not both (mutual exclusion), and "Paris" can only refer to a place if "Paris Hilton" is interpreted as hotel. We believe that a probabilistic database is well suited for such as task.

[1] http://geonames.org

2. Formal framework

The basis of this formalism is the possible world. We use the term possible world in the following sense: as long as the winning number has not been drawn yet in a lottery, you do not known the winner, but you can envision a possible world for each outcome. Analogously, one can envision multiple possible database states depending on whether certain facts are true or not. For example in Figure 1, a possible world (the true one) could contain annotations 1, 7, 8, and 10, but to a computer a world with annotations 2, 4, 5, and 6 could very well be possible too. Note that this differs from the use of the term 'possible world' in logics where it means possible interpretations [12, Chp.6] or as in modal logics [13].

The core of this formalization is the idea that we need to be able to identify the different possible worlds so we can reason about them. We do this by crafting a way to incrementally and constructively describe the name of a possible world.

2.1. Representation

Our formalization begins with the notion of a database as a possible world. A *database* $DB \in \mathbb{P}\, A$ consists of assertions $\{a_0, a_1, \ldots, a_n\}$ with a_i taken from A, the universe of assertions. 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. Since databases represent possible worlds, we use the symbols DB and w interchangeably. A probabilistic database PDB is a set of databases $\{DB_0, DB_1, \ldots, DB_n\}$, i.e., $PDB \in \mathbb{P}\,\mathbb{P}\, A$. Each different database represents a possible world in the probabilistic database. In other words, if an uncertainty is not distinguishable in the database state, i.e., if two databases are the same, then we regard this as one possible world. When we talk about possible worlds, we intend this to mean 'all possible worlds contained in the probabilistic database' denoted with W_{PDB}.

Implicit possible worlds Viewing it the other way around, an assertion holds in a subset of all possible worlds. To describe this relationship, we need an identification mechanism to refer to a subset of the possible worlds. For this purpose, we introduce the method of partitioning. A *partitioning* ω^n splits a database into n disjunctive parts each denoted with a *label* l of the form $\omega{=}v$ with $v \in 1..n$. If a world w is labelled with label l, we say that 'l holds for w.' Every introduced partitioning ω^n is a member of Ω, the set of introduced partitionings. W_l denotes the set of possible worlds in PDB labelled with l. $L(\omega^n) = \{\omega{=}v \mid v \in 1..n\}$ is the set of labels for partitioning ω^n.

In essence, possible worlds are about choices: choosing which assertions are in and which assertions are out. Independent choices may be composed, i.e.,

with k partitionings ω^n we obtain in the worst case n^k possible worlds.

Descriptive assertions and sentences A *descriptive assertion* is a tuple $\langle a, \varphi \rangle$ where φ is a *descriptive sentence*, a propositional formula describing how the assertion relates to the possible worlds where the partitioning labels of the form $\omega{=}v$ are the only type of atomic sentences. $\top$ denotes the empty sentence logically equivalent with *true* and $\bot$ the inconsistent sentence logically equivalent with *false*. The usual equivalence of sentences by equivalence of proposition formulae applies with the addition that $v_1 \neq v_2 \implies \omega{=}v_1 \wedge \omega{=}v_2 \equiv \bot$. Note that these descriptive sentences are a generalized form of the world set descriptors of MayBMS [14]. The functions $a(t)$ and $\varphi(t)$ denote the assertion and sentence component of tuple t, respectively. The evaluation function $W(\varphi)$ determines the set of possible worlds for which the sentence holds. It is inductively defined as

$$\begin{aligned} W(\omega{=}v) &= W_{\omega=v} \\ W(\varphi \vee \psi) &= W(\varphi) \cup W(\psi) \\ W(\varphi \wedge \psi) &= W(\varphi) \cap W(\psi) \\ W(\neg\varphi) &= W_{PDB} - W(\varphi) \\ W(\top) &= W_{PDB} \\ W(\bot) &= \varnothing \end{aligned}$$

Compact probabilistic database A *compact probabilistic database* is defined as a set of descriptive assertions and a set of partitionings: $CPDB = (D, \Omega)$. We consider *CPDB well-formed* iff all labels used in D are a member of Ω and all assertions are present only once hence with one descriptive sentence: $\forall t_1, t_2 \in D : t_1 \neq t_2 \implies a(t_1) \neq a(t_2)$. A non-well-formed compact probabilistic database can be made well-formed by reconstructing Ω from the labels used in D and merging the 'duplicate' tuples using the following transformation rules

$$\langle a, \varphi \rangle, \langle a, \psi \rangle \mapsto \langle a, \varphi \vee \psi \rangle \tag{1}$$

In general, φ denotes a set of possible worlds. The most restrictive set of worlds is described by a *fully described sentence* $\bar{\varphi}$ constructed as a conjunction of labels for each introduced partitioning of Ω. Because of well-formedness and because a possible world is only distinguished by the assertions it consists of, it follows that $\bar{\varphi}$ describes a single possible world. For example, given that $\Omega = \{\mathsf{x}^2, \mathsf{y}^3, \mathsf{z}^2\}$, one of the possible worlds is fully described by $\mathsf{x}{=}1 \wedge \mathsf{y}{=}2 \wedge \mathsf{z}{=}2$.

Let $L(\Omega)$ be the set of all possible fully described sentences: $L(\Omega) = \{l_1 \wedge \ldots \wedge l_k \mid \Omega = \{\omega_1^{n_1}, \ldots, \omega_k^{n_k}\} \wedge \forall i \in 1..k : l_i \in L(\omega_i^{n_i})\}$. The set of possible worlds contained in *CPDB* can now be defined as

$$W_{PDB} = \bigcup_{\varphi \in L(\Omega)} W(\varphi)$$

$$\begin{array}{ccccc} \{DB\} & \xleftarrow{f} & PDB & \xleftarrow{c} & CPDB \\ \downarrow{\scriptstyle \oplus} & & \downarrow{\scriptstyle \oplus} & & \downarrow{\scriptstyle \hat{\oplus}} \\ \{R\} & \xleftarrow{f} & PR & \xleftarrow{c} & CPR \end{array}$$

Figure 2: Commutative diagram illustrating the relationships between a set of databases, a probabilistic database, a compact probabilistic database and the associated query results.

Note that because each ω^n is a partitioning, the following holds

$$\forall \omega^n \in \Omega : W_{PDB} = \bigcup_{l \in L(\omega^n)} W_l$$

Dependencies Dependencies in the existence between assertions can be expressed with descriptive sentences logically combining different labels. *Mutual dependency* can be expressed by using the same sentence for the tuples. For example, $\langle a, \varphi \rangle$ and $\langle b, \varphi \rangle$ describes the situation where a and b both exist in a possible world or neither, but never only one of the two. *Implication* can be expressed by containment. For example, $\langle a, \varphi \rangle$ and $\langle b, \varphi \wedge \psi \rangle$ describes the situation that whenever a is contained in a possible world, then b is too. *Mutual exclusivity* can be expressed with mutually exclusive sentences, i.e., $\langle a, \varphi \rangle$ and $\langle b, \psi \rangle$ can never occur together in a possible world if $\varphi \wedge \psi \equiv \bot$.

Since each ω is a partitioning on its own, they can be considered as *independent* choices. For example, $\langle a, \mathsf{x}{=}1 \rangle$ and $\langle b, \mathsf{y}{=}1 \rangle$ use different partitionings, hence the labels establish no dependency between a and b and thus the existence of a and b is independent.

2.2. Querying

The concept of possible worlds means that querying a probabilistic database should be indistinguishable from querying each possible world separately, i.e., producing the same answers. This is illustrated in Figure 2 with a commutative diagram. The operations f and c represent formation and compaction, respectively. Formation constructs a probabilistic database from a set of databases. Compaction takes a probabilistic database and produces a compact probabilistic database. Both operations are trivially inverted as f' and c', through unpacking and enumerating all possible worlds, respectively.

For any query operator $\oplus$, we define an *extended operator* $\hat{\oplus}$ with an analogous meaning that operates on a compact representation. It is defined by $\hat{\oplus} = (\oplus, \tau_\oplus)$ where $\tau_\oplus$ 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 $\oplus$. We call an extended operator sound iff it adheres to the commutative

relations of Figure 2. This means, for example, that $\hat{\oplus} = (c \circ \oplus \circ c')$. Alternatively, starting from the non-compact probabilistic database *PDB*, the equality $(c \circ \oplus) = (\hat{\oplus} \circ c)$ must hold for any $\hat{\oplus}$.

Observe that we abstract from specific operators analogously to the way we abstract from the form of the actual data items. The above defines how to construct probabilistic operators from non-probabilistic ones. In this way, one can apply this to any query language in effect defining a family of probabilistic query languages.

2.3. Probability calculation

One can attach a probability $P(\omega{=}v)$ to each partition v of a partitioning ω^n provided that $\sum_{v=1}^{n} P(\omega{=}v) = 1$. As is known from the U-relations model [14] and variations thereof such as [1], calculating probabilities of possible worlds or the existence of an assertion among the worlds, can make use of certain properties that also apply here. For example, $P(\omega_1{=}v_1 \wedge \omega_2{=}v_2) = P(\omega_1{=}v_1) \times P(\omega_2{=}v_2)$ and $P(\omega_1{=}v_1 \vee \omega_2{=}v_2) = P(\omega_1{=}v_1) + P(\omega_2{=}v_2)$ iff $\omega_1 \neq \omega_2$. Moreover,

$$\begin{aligned} P(\langle a, \varphi \rangle) &= \sum_{\substack{w \in W_{PDB} \\ a \in w}} P(w) \\ &= \sum_{w \in W(\varphi)} P(w) \\ &= P(\varphi) \end{aligned}$$

Constraining the expressiveness of the descriptive sentences or requiring a normal form may allow for more efficient exact probability calculations, for example, [15] describes an efficient approach for calculating the probabilities of positive sentences in disjunctive normal form. Larger amounts of uncertainty, represented by large amounts of partitionings involved in the description of a possible world, may require approximate probability calculation to remain feasible. [16] details one such approach to this problem.

2.4. Comparison

The above-described framework is in essence a generalization of the U-relations model behind MayBMS [14]. Most other probabilistic database models [3, Chp.3] are also based on the concept of possible worlds. Our framework mainly distinguishes itself from these models on the following aspects

- We have abstracted from what the raw data looks like by treating them as assertions. In this way, we obtain data model independence whereas other models are defined for a specific data model.
- Our formal foundation is a framework turning a data model and query language into a probabilistic version, hence we have not defined one specific model but a family of models.
- The descriptive sentences represent the uncertainty metadata. As it is nicely separate from the raw data, we obtain a loose coupling between data and uncertainty metadata. This allows the development of a generic uncertainty management component that can be reused in systems using different data models. The uncertainty management functionality of existing prototypes is built into the probabilistic database itself and cannot easily be reused when developing another.
- Probabilities are separately attached as an 'optional add-on' obtaining the desired loose coupling between alternatives and probabilities.
- We allow full propositional logic for constructing descriptive sentences which results in an expressive mechanism for establishing complex dependencies. For probabilistic XML, [17] showed that PrXML families allowing cie nodes are fundamentally more expressive than the other families while these nodes only allow conjunctions of independent events whereas we allow any propositional sentence.

3. Illustration of data model independence

We illustrate the data model independence of our framework by applying it to (1) Datalog, and (2) Relational algebra

3.1. Framework applied to Datalog

Datalog is a knowledge representation and query language based on a subset of Prolog. It allows the expression of facts and rules. Rules specify how more facts can be derived from other facts. A set of facts and rules is known as a Datalog program.

In the sequel, we first define our Datalog language and then apply the framework to obtain probabilistic Datalog by viewing the facts and rules as assertions. We base our definition of Datalog on [12, Chp.6] (only positive Datalog for simplicity).

Definition of Datalog We postulate disjoint sets *Const*, *Var*, *Pred* as the sets of *constants*, *variables*, and *predicate symbols*, respectively. Let $c \in Const$, $X \in Var$, and $p \in Pred$. A *term* $t \in Term$ is either a constant or variable where $Term = Const \cup Var$.

An *atom* $A = p(t_1, \ldots, t_n)$ consists of an n-ary predicate symbol p and a list of argument terms t_i. An atom is *ground* iff $\forall i \in 1..n : t_i \in Const$. A *clause* or *rule* $r = (A^{\mathrm{h}} \leftarrow A_1, \ldots, A_m)$ is a horn clause representing the knowledge that A^{h} is true if all A_i are true. A *fact* is a rule without body $(A^{\mathrm{h}} \leftarrow)$. Let $vars(r)$ be the set of variables occurring in rule r. A set of rules KB is called a *knowledge base* or *program*. The usual safety conditions of pure Datalog apply.

An example of a Datalog program can be found below. It determines the country C of a phrase Ph

$$\varphi, \varphi_1, \ldots, \varphi_m \overset{\tau_\models}{\mapsto} \varphi \wedge \bigwedge_{i \in 1..m} \varphi_i$$

$$\frac{r \in \mathrm{KB} \quad r = (A^{\mathrm{h}} \leftarrow A_1, \ldots, A_m) \quad \exists \theta : A^{\mathrm{h}}\theta \text{ is ground} \wedge \forall i \in 1..m : \mathrm{KB} \models A_i\theta}{\mathrm{KB} \models A^{\mathrm{h}}\theta} \xrightarrow{\models} \frac{r \in \mathrm{KB} \quad r = (A^{\mathrm{h}} \overset{\varphi}{\leftarrow} A_1, \ldots, A_m) \quad \exists \theta : A^{\mathrm{h}}\theta \text{ is ground} \wedge \forall i \in 1..m : \mathrm{KB} \hat{\models} \langle A_i\theta, \varphi_i \rangle \quad \varphi' = \tau_\models(\varphi, \varphi_1, \ldots \varphi_m) \quad \varphi' \not\equiv \bot}{\mathrm{KB} \hat{\models} \langle A^{\mathrm{h}}\theta, \varphi' \rangle}$$

Figure 3: Definition of Datalog and application of our framework defining $\hat{\models}$ and $\tau_\models$ (base case with $m = 0$).

at position Pos if it is of type place and it refers to an entry in a gazetteer containing the country.

```
type(paris, pos1, place) ←
gazetteer(g11, paris, france) ←
refersto(paris, pos1, g11) ←

location(Ph, Pos, C) ←
   type(Ph, Pos, place), refersto(Ph, Pos, G),
   gazetteer(G, Ph, C)
```

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

Semantic entailment for our Datalog is defined in Figure 3 (left side of $\xrightarrow{\models}$) as the *Herbrand base*: all ground atoms that can be derived as a logical consequence from KB.

The three facts of our example are entailed directly, because their bodies are empty, hence $m = 0$, and the heads are already ground such that $\theta = \varnothing$ suffices. The location-rule contains variables. With $\theta = \{\mathsf{Ph/paris}, \mathsf{Pos/pos1}, \mathsf{G/g11}, \mathsf{C/france}\}$ or any superset thereof the atoms in the body turn into entailed facts allowing location(paris, pos1, france) to be entailed.

Probabilistic Datalog The approach to obtain Probabilistic Datalog using our framework is by viewing the facts and rules as assertions. We use the notation $(A^{\mathrm{h}} \overset{\varphi}{\leftarrow} A_1, \ldots, A_m)$ for the tuple $\langle A^{\mathrm{h}} \leftarrow A_1, \ldots, A_m, \varphi \rangle$. Note that this not only allows the specification of uncertain facts, but also uncertain rules as well as dependencies between the existence of facts and rules. In this way, the Probabilistic Datalog we obtain is more expressive than existing flavors of probabilistic Datalog such as ProbLog [18].

The 'operation' in Datalog is entailment. Therefore, applying our framework means defining probabilistic entailment $\hat{\models}$ by defining $\tau_\models$ and weaving it into the given definition of $\models$ (see Figure 3). The intuition behind the definition is that the descriptive sentence of an entailed fact is the conjunction of the sentences of the atoms and rules it is based on, which should not be 'false', i.e., it should not be equivalent to the sentence $\bot$.

Furthermore, probabilistic entailment needs to be well-formed. We achieve this by defining well-formed

```
type(paris_hilton, pos1-2, person) ←x=1
type(paris_hilton, pos1-2, hotel) ←x=2
type(paris, pos1, place) ←y=1 type(_, Pos, hotel), contains(pos1, Pos)
type(hilton, pos2, brand) ←z=1 type(_, Pos, hotel), contains(pos2, Pos)
gazetteer(g11, paris, france) ←⊤
gazetteer(g12, paris, canada) ←⊤
refersto(paris, pos1, g11) ←a=1
refersto(paris, pos1, g12) ←a=2

location(Ph, Pos, C) ←r=1
   type(Ph, Pos, place), refersto(Ph, Pos, G),
   gazetteer(G, Ph, C)
```

Figure 4: Example of a probabilistic Datalog program

entailment $\hat{\models}^*$ using transformation rule 1, i.e.,

$$\forall A \in \mathit{Atom} : \Phi_A \neq \varnothing \Rightarrow \mathrm{KB} \hat{\models}^* \langle A, \bigvee_{\varphi \in \Phi_A} \varphi \rangle$$
$$\text{where } \Phi_A = \{\varphi \mid \mathrm{KB} \hat{\models} \langle A, \varphi \rangle\}$$

Figure 4 contains an elaboration of our example in probabilistic Datalog. It expresses uncertainty about (a) whether "Paris Hilton" is person or a hotel, (b) whether "Paris" is a place and "Hilton" is a brand but only if they are part of a phrase that is interpreted as a hotel, (c) whether a phrase "Paris" refers to entry g11 or g12 in the gazetteer, and (d) whether or not our rule for determining the country is correct in general. Observe that both $\langle \mathsf{location(paris, pos1, france)}, \mathsf{r{=}1 \wedge y{=}1 \wedge x{=}2 \wedge a{=}1} \rangle$ and $\langle \mathsf{location(paris, pos1, canada)}, \mathsf{r{=}1 \wedge y{=}1 \wedge x{=}2 \wedge a{=}2} \rangle$ are entailed for this example.

Three kinds of (un)truth A language like probabilistic Datalog is an interesting vehicle to obtain deeper understanding of important concepts such as *truth* of facts that are uncertain. In fact, the language can express three kinds of untruth

1. A fact A is entailed with an inconsistent sentence $\varphi \equiv \bot$. This means that although A seems logically derivable, its derivation implies that the world is impossible, i.e., it is true in none of the possible worlds.
2. A fact A is entailed with a sentence φ with $P(\varphi) = 0$. This means that A is derived only for worlds with zero probability.
3. A fact A is not entailed (in any of the possible worlds). This is the original untruth of Datalog.

The differences between these untruths are rather subtle but nevertheless existing.

3.2. Framework applied to Relational Algebra

Relational Algebra is the underpinning of relational databases. It allows the expression of operations on data structured as relations containing tuples. The tuples in a relation are uniform and comply to the relation's schema which is defined as a set of attributes. The relations and tuples have a strong likeness to tables and rows known from SQL databases. Yet they are not equal: relations are sets of tuples, whereas tables in SQL are multisets.

Definition of Relational Algebra We postulate a set of *attribute domains Int*, *Bool*, *String*, etc. Let $R(at_1, \ldots, at_n) \subseteq \mathrm{dom}(at_1) \times \cdots \times \mathrm{dom}(at_n)$ be a *relation* containing *relational tuples* $r \in R$ with attributes $at_1, \ldots, at_n$ where $\mathrm{dom}(at_i)$ denotes the domain of at_i ($i \in 1..n$). Operations include the usual set operations *union* ($\cup$), *intersection* ($\cap$), and *difference* ($\setminus$) together with *selection* (σ), *projection* (π), *cartesian product* ($\times$), and *join* ($\bowtie$). The usual restrictions apply, for example, set operations require the operands to have the same attributes. We define the relational operators alongside the probabilistic ones below for easy comparison.

Probabilistic Relational Algebra Using our framework, we obtain probabilistic relational algebra by viewing relational tuples as assertions. For each operator $\oplus$, we define $\hat{\oplus}$ in terms of $\oplus$ and $\tau_\oplus$ where the latter maps descriptive sentences of the operands to a descriptive sentence of the result. We then 'weave' the application of $\tau_\oplus$ into the definition of the original non-probabilistic operators $\oplus$ (see Figure 5). Let $A(R) = \{a(t) \mid t \in R\}$ be the set of assertions (i.e., relational tuples) from a probabilistic relation R.

Note that we assume the probabilistic relational database as well as the result of every operation to be well-formed by applying transformation rule 1.

Figure 6 contains an example of the application of probabilistic relational algebra for our running example. Relation Type is an excerpt of Figure 1. Using relations RefersTo and Gazetteer we compute a new relation Locations with possible countries for the named entities:

$$\hat{\pi}_{\mathsf{phrase,pos,country}}(\hat{\sigma}_p(\mathsf{Type} \hat{\times} \mathsf{RefersTo} \hat{\times} \mathsf{Gazetteer}))$$

where $p = (\mathsf{Type.phrase} = \mathsf{RefersTo.phrase} \wedge \mathsf{Type.pos} = \mathsf{RefersTo.pos} \wedge \mathsf{RefersTo.gazetteer} = \mathsf{Gazetteer.id})$.

Type

phrase	pos	refers to	φ
Paris Hilton	1,2	person	x=1
Paris Hilton	1,2	hotel	x=2
Paris	1	place	y=1
Hilton	2	brand	z=1

Gazetteer

id	spelling	country	φ
g11	Paris	France	$\top$
g12	Paris	Canada	$\top$

RefersTo

phrase	pos	gazetteer	φ
Paris	1	g11	a=1
Paris	1	g12	a=2

Locations

phrase	pos	country	φ
Paris	1	France	y=1 $\wedge$ a=1 $\wedge$ $\top$
Paris	1	Canada	y=1 $\wedge$ a=2 $\wedge$ $\top$

Figure 6: Example relations with descriptive sentences. The 'Locations' relation is the result of $\hat{\pi}_{\mathsf{phrase,pos,country}}(\hat{\sigma}_p(\mathsf{Type} \hat{\times} \mathsf{RefersTo} \hat{\times} \mathsf{Gazetteer}))$.

4. Discussion

4.1. Optimizations

Scalable uncertainty A probabilistic database not only needs to be scalable in the volume of data, but also in the amount of uncertainty in the data. The latter presents itself both in the number of partitionings as well as in the size of the descriptive sentences. From our experience with a bio-informatics use case [6], the number of partitionings can easily grow into the thousands in real-world applications. The size of the descriptive sentences is determined by the complexity of the dependencies between assertions, its low-level representation, and allowed expressiveness.

Propositional logic techniques As propositional logic is the basis of the descriptive sentence, many algorithmic techniques can be applied. Equivalence-based sentence rewriting can be used for, e.g., simplification, normalization, and negation removal (a negated label can be substituted with an exhaustive disjunction of the other labels in the partitioning). An example of optimizations based on disjunctive normal form is [15]. Negation removal is particularly useful if the partitionings are restricted to be binary, which may be sufficient for certain applications and allows for many other optimizations. Another angle to consider is constraining the expressiveness of descriptive sentences which allows for optimization of its representation and manipulation.

During query execution, assertions with an inconsistent sentence can be filtered out. This, as well as the sentence rewriting techniques, can be done eagerly or lazily depending on the trade-off between overhead of the technique and resulting gains. Sentence manipulation can be optimized by taking into account properties of the operations, e.g., selection is guaranteed to produce a well-formed unmodified result, so no rewriting or filtering is necessary.

On the implementation level, special physical operators can combine data processing with sentence manipulation. For example, a merge-join implementation of $\hat{\bowtie}$ could combine joining tuples with simplification, normalization, and filtering.

$$\varphi \overset{\tau_\sigma}{\mapsto} \varphi \qquad \varphi, \psi \overset{\tau_\times}{\mapsto} \varphi \wedge \psi \qquad \varphi \overset{\tau_\pi}{\mapsto} \varphi \qquad \varphi \overset{\tau_\cup}{\mapsto} \varphi \qquad \varphi, \psi \overset{\tau_\cap}{\mapsto} \varphi \wedge \psi \qquad \varphi, \psi \overset{\tau_\setminus}{\mapsto} \varphi \wedge \neg\psi \qquad \varphi \overset{\tau_\setminus}{\mapsto} \varphi$$

$$\frac{r \in R \quad p(r)}{r \in \sigma_p(R)} \xrightarrow{\sigma} \frac{\varphi' = \tau_\sigma(\varphi) \quad \langle r, \varphi\rangle \in R \quad p(r)}{\langle r, \varphi'\rangle \in \hat{\sigma}_p(R)} \qquad \frac{r \in R \quad s \in S}{rs \in R \times S} \xrightarrow{\times} \frac{\varphi' = \tau_\times(\varphi, \psi) \quad \langle r, \varphi\rangle \in R \quad \langle s, \psi\rangle \in S}{\langle rs, \varphi'\rangle \in R \hat{\times} S}$$

$$\frac{r \in R(at_1, \ldots, at_n) \quad \{i_1, \ldots, i_k\} \in 1..n}{\langle r.at_{i_1}, \ldots, r.at_{i_k}\rangle \in \pi_{i_1..i_k}(R)} \xrightarrow{\pi} \frac{\varphi' = \tau_\pi(\varphi) \quad \{i_1, \ldots, i_k\} \in 1..n \quad \langle r, \varphi\rangle \in R(at_1, \ldots, at_n)}{\langle\langle r.at_{i_1}, \ldots, r.at_{i_k}\rangle, \varphi'\rangle \in \hat{\pi}_{i_1..i_k}(R)}$$

$$\frac{r \in R}{r \in R \cup S} \quad \frac{s \in S}{s \in R \cup S} \xrightarrow{\cup} \frac{\varphi' = \tau_\cup(\varphi) \quad \langle r, \varphi\rangle \in R}{\langle r, \varphi'\rangle \in R \cup S} \quad \frac{\psi' = \tau_\cup(\psi) \quad \langle s, \psi\rangle \in S}{\langle s, \psi'\rangle \in R \cup S}$$

$$\frac{r \in R \quad r \in S}{r \in R \cap S} \xrightarrow{\cap} \frac{\varphi' = \tau_\cap(\varphi, \psi) \quad \langle r, \varphi\rangle \in R \quad \langle r, \psi\rangle \in S}{\langle r, \varphi'\rangle \in R \hat{\cap} S}$$

$$\frac{r \in R \quad r \notin S}{r \in R \setminus S} \xrightarrow{\setminus} \frac{\varphi' = \tau_\setminus(\varphi, \psi) \quad \langle r, \varphi\rangle \in R \quad \langle r, \psi\rangle \in S}{\langle r, \varphi'\rangle \in R \hat{\setminus} S} \quad \frac{\varphi' = \tau_\setminus(\varphi) \quad \langle r, \varphi\rangle \in R \quad r \notin A(S)}{\langle r, \varphi'\rangle \in R \hat{\setminus} S}$$

Figure 5: Definitions of $\tau_\oplus$ and $\hat{\oplus}$ for probabilistic relational algebra. $\hat{\bowtie}_p \equiv \hat{\sigma}_p \circ \hat{\times}$.

Constraining expressiveness The full expressiveness of propositional logic allows for the expression of rich dependencies between assertions at the price of computational complexity. Restricting expressiveness can provide optimization benefits, e.g., disallowing negation may allow many optimizations that are not valid in its presence.

The data model and query language may already place lower requirements on the expressiveness of the descriptive sentences. For example, the only logical connective in probabilistic Datalog of Section 3.1 is conjunction, and disjunction is necessary for maintaining well-formedness. Hence, negation is not needed and also conjunction and disjunction only appear in particular patterns. Vice versa, restrictions on the descriptive sentences may restrict the query language as well. For example, without negation the difference between relations cannot be supported in probabilistic relational algebra.

4.2. Open problems

Alternative data models We have shown how our framework can be applied to Datalog and relational algebra. It seems equally possible to apply it to other data models such as graph, XML, and NoSQL types of databases. Opportunities still exist in the well-researched area of probabilistic relational databases. E.g., in column stores such as MonetDB [19], a relation is a set of columns; by also viewing columns as assertions, schema uncertainty as a result of schema integration [2] can naturally be supported. A data model's properties also allows special optimizations, e.g., in XML implicit dependencies between parent and child nodes can be exploited for optimization.

Efficient probability calculation Calculation of exact probabilities for query results may be computationally expensive and even exceed processing of the query itself. This cost can be mitigated by (1) only calculating probabilities on-demand such as in Trio [7], (2) approximating probabilities typically given some error bound, (3) caching probability calculation results for long shared parts of frequently occurring descriptive sentences. Furthermore, applying simpler probabilistic models also allows for more efficient probability calculation, exact or approximate.

Aggregates Figure 2 determines the semantics of traditional aggregates such as SUM (Σ): $\hat{\Sigma} = (c' \circ \Sigma \circ c)$. The difference with the other relational operators is that their direct computation over a compact probabilistic database is much less straightforward, because they may produce an answer that exponentially grows with growing numbers of partitionings. For example, given a probabilistic relation $R = \{\langle 1, \mathsf{x}{=}1\rangle, \langle 2, \mathsf{x}{=}1 \vee \mathsf{y}{=}1\rangle, \langle 3, \mathsf{x}{=}2 \wedge \mathsf{z}{=}1\rangle, \langle 5, \mathsf{y}{=}2\rangle\}$ with $\Omega = \{\mathsf{x}^2, \mathsf{y}^2, \mathsf{z}^2\}$, the answer of $\hat{\Sigma}(R)$ is $\{\langle 2, \mathsf{x}{=}2 \wedge \mathsf{y}{=}1 \wedge \mathsf{z}{=}2\rangle, \langle 3, \mathsf{x}{=}1 \wedge \mathsf{y}{=}1\rangle, \langle 5, \mathsf{x}{=}2 \wedge ((\mathsf{y}{=}1 \wedge \mathsf{z}{=}1) \vee (\mathsf{y}{=}2 \wedge \mathsf{z}{=}2)\rangle, \langle 8, \mathsf{y}{=}2 \wedge (\mathsf{x}{=}1 \vee (\mathsf{x}{=}2 \wedge \mathsf{z}{=}1))\rangle\}$. Observe that although not every possible world results in a different answer, it is an open problem how to construct sentences for the answers in an efficient way, i.e., without enumerating worlds.

Note, however, that in many applications it is not necessary to determine the full set of possible exact answers with their probabilities. [20] proposes a variety of answer forms for aggregate queries that can be (more) efficiently computed and may still be sufficiently informative such as (a) a single value

representing the expected value of the sum, (b) two values representing the mean of the sum and its standard deviation, (c) a histogram with probabilities for a predetermined number of answer ranges, (d) a single answer representing the single most likely value possibly with its probability, or (e) a top-k of the k most likely results, and so forth.

Out-of-world aggregations Many systems offer the expected value as an aggregation function. Furthermore, whereas computing a sum over probabilistic data has exponential complexity, computing the expected value of a sum has not. Therefore, such systems offer combined aggregators such as the 'esum'. This poses the questions of: are these truly aggregators; and what is an aggregate really?

Traditional aggregates operate by aggregating values over a dimension, possibly in groups, where a dimension typically is an attribute of a relation. The possible worlds can be seen as yet another dimension. For this reason, the expected value is indeed an aggregator, namely one operating over the possible worlds dimension. This insight has the potential of treating all aggregates, including the probabilisticly inspired ones, uniformly as well as combinations of aggregators. Note also that asking for the probability of a tuple or for an expected value forms a new class of query operators: they have no counterpart in the non-probabilistic query language. More research is needed to explore the implications of this new class of queries.

5. Conclusions

We revisited the formal foundations of probabilistic databases by proposing a formal framework that is based on attaching a propositional logic sentence to data assertions to describe the possible worlds in which that assertion holds. By doing so, the formalisation (a) abstracts from the underlying data model obtaining data model independence, and (b) separates metadata on uncertainty and probabilities from the raw data.

Data model independence of the framework is validated by applying it to Datalog and relational algebra to obtain probabilistic variants thereof: for every query operator $\oplus$, we define (a) sentence manipulation function $\tau_{\oplus}$ and (b) probabilistic query operator $\hat{\oplus}$, the latter by weaving $\tau_{\oplus}$ into the original definition of $\oplus$.

In relation to the framework, we discuss open problems such as alternative data models, probability calculation, and aggregation, as well as scalability and optimization issues brought to light due to the framework's properties.

References

[1] M. van Keulen. Managing uncertainty: The road towards better data interoperability. *J. IT - Information Technology*, 54(3):138–146, May 2012.

[2] M. Magnani and D. Montesi. A survey on uncertainty management in data integration. *J. Data and Information Quality*, 2(1):5:1–5:33, 2010. ISSN 1936-1955.

[3] F. Panse. *Duplicate Detection in Probabilistic Relational Databases*. PhD thesis, University of Hamburg, Germany, December 2014.

[4] F. Panse, M. van Keulen, and N. Ritter. Indeterministic handling of uncertain decisions in deduplication. *J. Data and Information Quality*, 4(2):9:1–9:25, March 2013.

[5] M. van Keulen and A. de Keijzer. Qualitative effects of knowledge rules and user feedback in probabilistic data integration. *The VLDB Journal*, 18(5):1191–1217, 2009.

[6] B. Wanders, M. van Keulen, and P.E. van der Vet. Uncertain groupings: probabilistic combination of grouping data. Technical Report TR-CTIT-14-12, Centre for Telematics and Information Technology, University of Twente, Enschede, 2014.

[7] J. Widom. Trio: A system for integrated management of data, accuracy, and lineage. In *CIDR*, pages 262–276, 2005.

[8] C. Koch. MayBMS: A system for managing large probabilistic databases. *Managing and Mining Uncertain Data*, pages 149–183, 2009.

[9] D. Olteanu, Jiewen Huang, and C. Koch. SPROUT: Lazy vs. eager query plans for tuple-independent probabilistic databases. In *Proc. of ICDE*, pages 640–651, March 2009.

[10] M. van Keulen and M.B. Habib. Handling uncertainty in information extraction. In *Proc. of URSW*, volume 778 of *CEUR Workshop Proceedings*, pages 109–112, October 2011. ISSN 1613-0073.

[11] J. Kuperus, C. Veenman, and M. van Keulen. Increasing NER recall with minimal precision loss. In *Proc. of EISIC*, pages 106–111, August 2013. ISBN 978-0-7695-5062-6.

[12] S. Ceri, G. Gottlob, and L. Tanca. *Logic Programming and Databases*. Springer, 1990. ISBN 3-540-51728-6.

[13] N.B. Cocchiarella and M.A. Freund. *Modal Logic: An Introduction to its Syntax and Semantics*. Oxford University Press, August 2008. ISBN 978-0195366570.

[14] L. Antova, T. Jansen, C. Koch, and D. Olteanu. Fast and simple relational processing of uncertain data. In *Proc. of ICDE*, pages 983–992, 2008.

[15] C. Koch and D. Olteanu. Conditioning probabilistic databases. *Proc. of the VLDB*, 1(1):313–325, 2008.

[16] D. Olteanu, J. Huang, and C. Koch. Approximate confidence computation in probabilistic databases. In *Proc. of ICDE*, pages 145–156, 2010.

[17] S. Abiteboul, B. Kimelfeld, Y. Sagiv, and P. Senellart. On the expressiveness of probabilistic XML models. *The VLDB Journal*, 18(5):1041–1064, 2009.

[18] L. De Raedt, A. Kimmig, and H. Toivonen. ProbLog: A probabilistic Prolog and its application in link discovery. *IJCAI*, 7:2462–2467, 2007.

[19] P.A. Boncz, S. Manegold, and M.L. Kersten. Database architecture evolution: Mammals flourished long before dinosaurs became extinct. *Proc. of the VLDB*, 2(2):1648–1653, 2009.

[20] D. Knippers. Querying uncertain data in xml. Master's thesis, University of Twente, August 2014. http://purl.utwente.nl/essays/65632.