ILLINOIS INSTITUTE OF TECHNOLOGY

Uncertainty Annotated Databases - A Lightweight Approach for Dealing with Uncertainty

Su Feng, Aaron Huber, Boris Glavic, Oliver Kennedy

IIT DB Group Technical Report
IIT/CS-DB-2018-01

2018-02

http://www.cs.iit.edu/~dbgroup/

Uncertainty Annotated Databases - A Lightweight Approach for Dealing with Uncertainty (technical report)

Su Feng
Illinois Institute of Technology
sfeng14@hawk.iit.edu

Aaron Huber
SUNY Buffalo
ahuber@buffalo.edu

Boris Glavic
Illinois Institute of Technology
bglavic@iit.edu

Oliver Kennedy
SUNY Buffalo
okennedy@buffalo.edu

***Abstract*—Incomplete data models have been proposed to deal with the uncertainty inherent in many real world data collection and management tasks. However, query evaluation over such models, i.e., computing which potential query answers are certain, is of high computational complexity. We introduce UA-databases (UA-DBs), a light-weight model of uncertainty where tuples from a single possible world are annotated with uncertainty information. Importantly, queries over UA-DBs preserve the annotated possible world and return an under-approximation of certain answers. UA-DBs can be derived from common incomplete and probabilistic data models. We implement our approach on top of a DBMS and experimentally demonstrate its efficiency. Furthermore, we show that for many real world datasets the amount of certain answers which are misclassified as uncertain by our approach is low.**

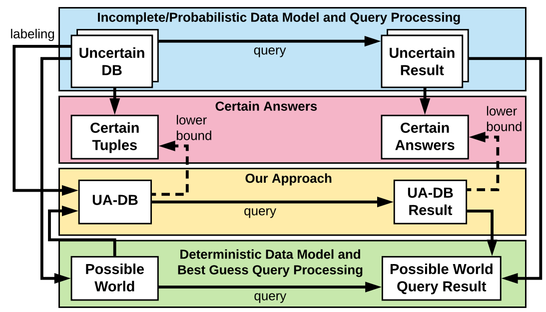


Fig. 1: UA-DBs relative to Deterministic, Incomplete/Probabilistic, and Certain Answer query processing schemes.

I. Introduction

Data uncertainty arises naturally in applications like sensing [28], data exchange [12], distributed computing [27], constraint-based data cleaning [8], and many others. At present, users spend significant effort to curate their data upfront to avoid having to deal with uncertainty during analysis. Incomplete [23] and probabilistic databases [36] have emerged as a principled way of dealing with uncertainty. Both types of databases consist of a set of deterministic instances called possible worlds that represent possible interpretations of the real world. When processing queries over uncertain data, we may want to know which query results are *certain answers* [2], [23], i.e., they are in the result, independent of which possible world matches the real world. For probabilistic databases, a related problem is to compute the *marginal probability* of an answer being in the result. However, the computational complexity of such semantics is high. For example, computing the certain answers for first-order queries is coNP-complete [2], [23] (data complexity). As a result, these techniques are rarely used in practice. Instead, users resort to *best guess* query processing, in effect making an educated guess about which possible world to use and then working exclusively with this world.

In this paper, we propose a principled, easy-to-use, and light-weight approach for representing and querying uncertain data called *Uncertainty-Annotated Databases* (*UA-DBs*). Figure 1 shows an overview of our approach. A UA-DB is a single possible world from an incomplete or probabilistic database, such as the one obtained from curation, where tuples are labeled as being either certain or uncertain. Queries over a UA-DB propagate these annotations through lightweight "extensional" semantics [36]. Thus, a UA-DB can be implemented through simple query rewriting on top of any classical relational database system with minimal performance overhead compared to deterministic query processing. We show how many existing models of uncertainty including Tuple-Independent Databases [36], x-databases [3], and C-Tables [23] can be converted into UA-DBs. Notably, the result of a query Q over a UA-DB bounds the set of certain answers of Q from above and below. An upper bound is provided through the possible world encoded by a UA-DB, since the result of a query over a possible world contains all certain answers. Furthermore, the set of tuples that a UA-DB marks as certain bounds the certain answers from below. We show, both analytically and experimentally, that this lower bound is a good approximation. Under specific conditions, exactly the set of certain answers will be marked as certain. We show experimentally that even when these conditions do not hold, the amount of misclassified certain answers is low.

Example 1. *Geocoders translate natural language descriptions of locations into coordinates (i.e., latitude and longitude). Consider the following example locations* [1] *in the Buffalo area, and some possible geocodings:*

[1] http://projects.buffalonews.com/charts/shootings/index.html

id	locale	state	p
1	Lasalle	NY	**1.0**
2	Tucson	AZ	**0.8**
2	Grant Ferry	NY	0.2
3	Kingsley	NY	**1.0**
4	Kensington	NY	**1.0**

(a) Incomplete/Probabilistic Query Processing

id	locale	state
1	Lasalle	NY
3	Kingsley	NY
4	Kensington	NY

(b) Certain Answers

id	locale	state	certain?
1	Lasalle	NY	Certain
2	Tucson	AZ	Uncertain
3	Kingsley	NY	Uncertain
4	Kensington	NY	Certain

(c) Uncertainty Annotated Database

Fig. 2: Examples of query results under different evaluation semantics over uncertain data.

id	address	geocoded
1	51 Comstock	(42.93, -78.81)
2	Grant at Ferguson	(32.25, -110.87) or (42.91, -78.89)
3	499 Woodlawn	(42.91, -78.84) or (42.91, -78.84)
4	192 Davidson	(42.93, -78.80)

Figure 2 shows the output of a spatial join that determines the neighborhood (attribute `locale`*) of each set of coordinates. Using standard probabilistic query semantics (Figure 2a), queries over uncertain data produce the set of all* possible *outputs. Each tuple is annotated with its confidence: the marginal probability that the tuple appears in the result. Conversely, the set of certain answers (Figure 2b) is the set of tuples that are guaranteed to appear in the result. In the former case, the result is large, as all possible outcomes are listed. Conversely, in the latter case, some (potentially important) result records do not appear at all. Figure 2c shows the result of the same query as a UA-DB. The result given here is the set of tuples corresponding to the possible world containing the most likely geocodings for each location. Tuples labeled as certain (e.g., locations 1 and 4) are guaranteed to be part of the query's answer across all possible worlds. Tuples labeled as uncertain may, in some possible worlds, be absent from the result (e.g., location 2). UA-DBs are a conservative approximation of certain answers and it is possible that some tuples labeled as uncertain will actually be present in all possible worlds (e.g. location 3). However, even if a certain answer is mislabeled as uncertain, it is still present in the result of the UA-DB query.*

There is previous work on under-approximating certain answers [30], [16], [34], based on the view that missing a certain answer is better than mis-reporting confidence. Our approach significantly improves this state-of-the-art: 1) Being based on Green et. al.'s semiring annotation framework [18], UA-DBs generalize the classical *set-oriented* notion of incomplete data and certain answers, to any data model expressible through relations annotated with elements from an l-semiring [26], [18] — a semiring that has a well-behaved greatest lower bound operation. Thus, apart from [37] and [21], to the best of our knowledge we are the first to generalize certain answers and incomplete databases beyond set semantics. Although we mainly focus on set- and bag-relational data models in this paper, many of our results are of much broader applicability. For instance, they apply to data with uncertain provenance or data with uncertain access control levels; 2) We split the computation of certain answers into two subproblems: under-approximating the set of certain tuples in an incomplete database and preserving this under-approximation when evaluating queries. Thus, in contrast to previous approaches that focus on specific encodings of incomplete data such as V-tables [23], we support a wide range of incomplete and probabilistic data representations; 3) In contrast to prior work on purely under-approximating certain answers, UA-DB query results are guaranteed to include all certain answers, even though some may be mislabeled.

A. Outline

The remainder of the paper is organized as follows.

Incomplete $\mathcal{K}$-Relations. (Section III) We present a formal model for incomplete databases based on $\mathcal{K}$-relations [18]. We define a novel "pivoted" encoding of incomplete $\mathcal{K}$-relations called $\mathcal{K}_W$-relations and prove that queries over such relations realize possible worlds query semantics. Furthermore, we generalize the classical notion of certain answers to $\mathcal{K}_W$-relations based on the observation that certain answers represent a lower bound on the content of a possible world. It is thus natural to define certainty based on a greatest-lower-bound operation (GLB) for semiring annotations based on so-called l-semirings [26] where the GLB is well behaved. We show that certainty in our model corresponds to the classical definition of certain answers for both set [31] and bag [22] semantics.

Approximating Certain Answers. (Section IV) We define *uncertainty labelings*, which are $\mathcal{K}$-relations that under-approximate the set of certain tuples for $\mathcal{K}_W$-relations. An uncertainty labeling is certain- or *c-sound* (resp., *c-complete*) if it is a lower (resp., upper) bound on the set of certain tuples in a $\mathcal{K}_W$-relation; and *c-correct* if it is both. We also extend these definitions to query semantics. A query semantics preserves c-soundness if the result of the query is a c-sound labeling for the result of evaluating the query over the input $\mathcal{K}_W$-database from which the labeling was derived.

Creating Uncertainty Labelings. (Section V) We show how to derive labelings for common uncertain data encodings including tuple-independent databases, x-relations, and C-tables. For the former two we prove that the labeling is guaranteed to be c-complete, while for the latter we prove that the labeling is guaranteed to be at least c-sound.

Queries over Uncertainty Labelings. (Section VI) Since labelings are $\mathcal{K}$-relations, we can evaluate queries over such labelings. We demonstrate that evaluating queries in this fashion preserves the under-approximation of certain answers. That is queries preserve c-soundness. Furthermore, under certain conditions this query semantics returns precisely the certain answers. That is, since all queries preserve c-soundness, under these conditions queries preserve c-correctness.

UA-DBs. (Section VII) We then define an *UA-DB* to be a possible world from an incomplete/probabilistic database annotated with an uncertainty labeling and prove that queries over UA-DBs preserve both the possible world and labeling.

Implementation for Bag Semantics. (Section VIII) We implement UA-DBs on top of a standard relational DBMS. We extend the schema of relations with an additional attribute to label tuples as either certain or uncertain (e.g., Figure 2c). Queries with UA-relational semantics are compiled into standard relational queries over this encoding. We prove this compilation process to be correct.

Performance. In Section X, we demonstrate experimentally that UA-DBs outperform state of the art probabilistic/incomplete query processing schemes and are competitive with deterministic query evaluation. We also show empirically that for a wide range of real world datasets, the amount of certain answers misclassified by our approach as uncertain is low.

II. NOTATION AND BACKGROUND

A database schema $\mathbf{D} = \{\mathbf{R}_1, \ldots, \mathbf{R}_n\}$ is a set of relation schemas. A relational schema $\mathbf{R}(A_1, \ldots, A_n)$ consists of a relation name and a set of attribute names $A_1, \ldots, A_n$. The arity $arity(\mathbf{R})$ of a relation schema $\mathbf{R}$ is the number of attributes in $\mathbf{R}$. A database instance D for database schema $\mathbf{D}$ is a set of relation instances with one relation for each relation schema in $\mathbf{D}$: $D = \{R_1, \ldots, R_n\}$. Assume a universal domain of attribute values $\mathbb{D}$. A tuple with schema $\mathbf{R}$ is an element from $\mathbb{D}^{arity(\mathbf{R})}$. In this paper, we consider both bag and set semantics. A set relation R with schema $\mathbf{R}$ is a set of tuples with schema $\mathbf{R}$, i.e., $R \subseteq \mathbb{D}^{arity(\mathbf{R})}$. A bag relation R with schema $\mathbf{R}$ is a bag (multiset) of tuples with schema $\mathbf{R}$. We use TUPDOM to denote the set of all tuples over domain $\mathbb{D}$.

A. Possible Worlds Semantics

Incomplete and probabilistic databases model uncertainty and its impact on query results. An *incomplete database* $\mathcal{D}$ is a set of deterministic database instances $D_1, \ldots, D_n$ called *possible worlds*. We write $t \in D$ to denote that a tuple t appears in a specific possible world D. A *probabilistic database* is an incomplete database paired with a probability distribution $P : \{D_i\} \to [0, 1]$ over possible worlds. In the following will mostly focus on incomplete databases.

Example 2. *Continuing Example 1, Figure 3 shows the two possible worlds encoded by the instance from Figure 2a. Observe that some tuples (e.g.,* ⟨ 1, Lasalle, NY ⟩*) appear in all possible worlds — these are part of the instance no matter which possible world is chosen. Such tuples are called* ***certain****. Tuples that appear in at least one possible world (e.g.,* ⟨ 2, Tuscon, AZ ⟩*) are called* ***possible****.*

Decades of research [36], [23], [20], [5], [6], [3] has focused on algorithms for efficient query processing over incomplete or probabilistic databases. These techniques commonly adopt the

id	locale	state
1	Lasalle	NY
2	**Tucson**	**AZ**
3	Kingsley	NY
4	Kensington	NY

(a) D_1

id	locale	state
1	Lasalle	NY
2	**Grant Ferry**	**NY**
3	Kingsley	NY
4	Kensington	NY

(b) D_2

Fig. 3: An example incomplete database $\mathcal{D} = \{D_1, D_2\}$.

id	locale	state
1	Lasalle	NY
3	Kingsley	NY
4	Kensington	NY

(a) $Q_{NY}(D_1)$

id	locale	state
1	Lasalle	NY
2	Grant Ferry	NY
3	Kingsley	NY
4	Kensington	NY

(b) $Q_{NY}(D_2)$

Fig. 4: The result of a query Q over an incomplete database $\mathcal{D}$ is a set of possible worlds derived by evaluating Q over each possible world $D \in \mathcal{D}$.

so called "possible worlds" semantics: The result of evaluating a query Q over an incomplete database is the set of relation instances resulting from evaluating Q over each possible world individually using standard deterministic query semantics.

$$Q(\mathcal{D}) = \{\ Q(D) \mid D \in \mathcal{D}\ \} \tag{1}$$

Example 3. *Consider the following query* $Q_{NY} = \sigma_{state='NY'}(\mathcal{D})$*, which returns locations in NY State from the database* $\mathcal{D}$ *shown in Figure 3. The result of* $Q_{NY}(\mathcal{D})$ *is the set of worlds computed by evaluating* Q_{NY} *over each world of* $\mathcal{D}$ *as shown in Figure 4. Observe that the row for location 2 appears in* $Q_{NY}(D_2)$*, but not* $Q_{NY}(D_1)$*.*

B. Certain, Possible, and Best-Guess Answers

The goal of query processing over incomplete databases is to differentiate query results that are certain from ones that are merely possible. Formally, a tuple is certain if it appears in every possible world[2] [23], [31]:

$$certain(\mathcal{D}) = \{t \mid \forall D \in \mathcal{D} : t \in D\} \tag{2}$$

$$possible(\mathcal{D}) = \{t \mid \exists D \in \mathcal{D} : t \in D\} \tag{3}$$

In contrast to [23], which studies certain answers to queries, we define certainty at the instance level. The two approaches are equivalent since we can compute the certain answers of query Q over incomplete instance $\mathcal{D}$ as $certain(Q(\mathcal{D}))$. Although computing certain answers is coNP-hard [2] in general, there exist PTIME approximations [30], [21], [34] that allow for false positives (tuples are returned that are not certain answers) or false negatives (some certain answers may be omitted from the query result).

Best Guess Queries. As mentioned in the introduction, another approach commonly used in practice is to ignore ambiguity in the data and select one possible world. Queries are evaluated solely in this world, and ambiguity is ignored or documented outside of the database. We refer to this approach as *best-guess query processing* (BGQP) [39] since typically one would like to select the possible world that is deemed most

[2]In probabilistic databases, certain tuples appear in worlds summing to a probability of 1. Without loss of generality we assume that this only happens if the tuple appears in all possible wolds.

id	address	ℓ	$\mathbb{N}$	$\mathbb{B}$
1	51 Co...	L_1	1	T
2	Grant ...	L_2	1	T
3	499 W...	L_4	1	T

(a) Address

ℓ	locale	state	$\mathbb{N}$	$\mathbb{B}$
L_1	L...	NY	1	T
L_2	T...	AZ	1	T
L_3	G...	NY	1	T
L_4	K...	NY	1	T
L_5	W...	IL	1	T

(b) Neighborhood

state	$\mathbb{N}$	$\mathbb{B}$
NY	$2 = (1 \cdot 1) + (1 \cdot 1)$	$T = (T \wedge T) \vee (T \wedge T)$
AZ	$1 = (1 \cdot 1)$	$T = (T \wedge T)$
IL	$0 = (0 \cdot 1)$	$F = (F \wedge T)$

(c) Result of Q_a

Fig. 5: $\mathbb{N}$- and $\mathbb{B}$-relation examples

likely (e.g., the world with the highest probability). We refer to the selected world as the *best guess world* BESTGUESS($\mathcal{D}$).

C. K-relations

Our generalization of incomplete databases, uncertainty labeling schemes, and UA-relations are all based on the $\mathcal{K}$-**relation** [18] framework. In this framework, relations are annotated with elements from the domain K of a commutative semiring $\mathcal{K}$. A structure $\mathcal{K} = (K, \oplus_{\mathcal{K}}, \otimes_{\mathcal{K}}, \mathbb{1}_{\mathcal{K}}, \mathbb{0}_{\mathcal{K}})$ is a commutative semiring iff (1) addition and multiplication are associative and commutative, (2) $\mathbb{0}_{\mathcal{K}}$ is the neutral element of addition, (3) $\mathbb{1}_{\mathcal{K}}$ is the neutral element of multiplication, (4) multiplication with $\mathbb{0}_{\mathcal{K}}$ yields $\mathbb{0}_{\mathcal{K}}$, and (5) multiplication distributes over addition. For simplicity, we will refer to commutative semirings as semirings.

As before, we use $\mathbb{D}$ to denote a universal domain of values. An n-nary $\mathcal{K}$-relation is a function that maps tuples (elements from $\mathbb{D}^n$) to elements from K. Tuples that are not in the relation are annotated with $\mathbb{0}_{\mathcal{K}}$. If $\mathbb{D}$ is infinite, then it is additionally required that only finitely many tuples are mapped to an element other than $\mathbb{0}_{\mathcal{K}}$ (i.e., relations must be finite). What type of information is encoded by the annotations assigned to tuples depends on the semiring $\mathcal{K}$ that is used.

Encoding Sets and Bags. Consider the following two examples of commutative semirings: the natural numbers ($\mathbb{N}$) with addition and multiplication $(\mathbb{N}, +, \times, 0, 1)$ and boolean constants $\mathbb{B} = \{T, F\}$ with disjunction ($\vee$) and conjunction ($\wedge$) forming the semiring $(\mathbb{B}, \vee, \wedge, F, T)$. In the following, we will abuse notation and denote by $\mathbb{N}$ and $\mathbb{B}$ respectively, both the domain and the corresponding semiring structure. Green et al. [18] demonstrated that, for an appropriate choice of semiring, $\mathcal{K}$-relations can be used to encode standard set and bag semantics, as well as many extensions of the relational model including various types of provenance. Specifically, $\mathbb{B}$ ($\mathbb{N}$) encodes set (bag) semantics by annotating all tuples in the relation with true (their multiplicity) and all other tuples with false (0). Since $\mathcal{K}$-relations are functions from tuples to annotations, it is customary to denote the annotation of a tuple t in relation R as $R(t)$ (applying function R to input t).

Query Semantics. Operators of the positive relational algebra ($\mathcal{RA}^+$) over $\mathcal{K}$-relations are defined by combining input annotations using operations $\oplus_{\mathcal{K}}$ and $\otimes_{\mathcal{K}}$.

$$\textbf{Union: } [R_1 \cup R_2](t) = R_1(t) \oplus_{\mathcal{K}} R_2(t)$$
$$\textbf{Join: } [R_1 \bowtie R_2](t) = R_1(t[R_1]) \otimes_{\mathcal{K}} R_2(t[R_2])$$
$$\textbf{Projection: } [\pi_U(R)](t) = \sum_{t=t'[U]} R(t')$$
$$\textbf{Selection: } [\sigma_\theta(R)](t) = R(t) \otimes_{\mathcal{K}} \theta(t)$$

For simplicity we assume in the definition above that tuples are of a compatible schema (e.g., $\mathbf{R}_1$ for a union $R_1 \cup R_2$). We use $\theta(t)$ to denote a function that returns $\mathbb{1}_{\mathcal{K}}$ iff θ evaluates to true over the values from tuple t and $\mathbb{0}_{\mathcal{K}}$ otherwise, and $t[U]$ and $t[R]$ to denote projecting tuple t on a set of attributes U and restricting it to the schema of relation R, respectively.

Example 4. *Figure 5 shows an example of how to encode a bag semantics database as an $\mathbb{N}$-database by annotating each tuple t with its multiplicity (the number of duplicates of t that exist in the relation). Annotations are shown to the right of each tuple. Query Q_a, shown below, returns states.*

$$Q_a = \pi_{state}(\textsf{Address} \bowtie \textsf{Neighborhood})$$

In the input database every tuple appears once (is annotated with 1). The annotation of an output tuple is computed by multiplying annotations of joined tuples, and summing up annotations projected onto the same result tuple. For instance, 2 NY addresses are returned.

In the following, we will make use of homomorphisms. A mapping $h : \mathcal{K} \to \mathcal{K}'$ from a semiring $\mathcal{K}$ to a semiring $\mathcal{K}'$ is a called a **homomorphism** if for any $k, k' \in K$:

$$h(\mathbb{0}_{\mathcal{K}}) = \mathbb{0}_{\mathcal{K}'} \qquad h(\mathbb{1}_{\mathcal{K}}) = \mathbb{1}_{\mathcal{K}'}$$
$$h(k \oplus_{\mathcal{K}} k') = h(k) \oplus_{\mathcal{K}'} h(k') \quad h(k \otimes_{\mathcal{K}} k') = h(k) \otimes_{\mathcal{K}'} h(k')$$

As observed by Green et al. [18], any semiring homomorphism h can be lifted to a homomorphism from $\mathcal{K}$-relations to $\mathcal{K}'$-relations by applying h to the annotation of every tuple t: $h(R)(t) = h(R(t))$. Importantly, it was shown that queries commute with semiring homomorphisms. That is, given a homomorphism h, query Q, and $\mathcal{K}$-database D we have $h(Q(D)) = Q(h(D))$. We will abuse syntax and use the same function symbols (e.g., $h(\cdot)$) to denote mappings between semirings, $\mathcal{K}$-relations, as well as $\mathcal{K}$-databases.

Example 5. *Continuing with Example 4, assume we want set semantics instead. We can derive a set semantics instance (semiring $\mathbb{B}$) by applying to each table a mapping $h : \mathbb{N} \to \mathbb{B}$ defined as $h(k) = T$ if $k > 0$ and $h(k) = F$ otherwise. The mapping h is a semiring homomorphism (we leave the proof as an exercise for the reader). Thus, evaluating Q_a in $\mathbb{N}$ and then applying h to the result (i.e., $h(Q(D))$) produces the same result as applying h to the input database and then evaluating Q_a in $\mathbb{B}$ (i.e., $Q(h(D))$).*

When defining upper and lower bounds for annotations in Section III, we will make use of the fact that for many semirings $\mathcal{K}$ it is possible to define an order relation over their domain K based on the addition operation $\oplus_{\mathcal{K}}$. This order relation is called the *natural order*. The natural order $\preceq_{\mathcal{K}}$ for a semiring $\mathcal{K}$ is defined as: An element k precedes k' if it is possible to obtain k' by adding any value k'' to k.

$$\forall k, k' \in K : k \preceq_{\mathcal{K}} k' \Leftrightarrow \exists k'' \in K : k \oplus_{\mathcal{K}} k'' = k' \qquad (4)$$

Semirings for which the natural order is a partial order are called *naturally ordered* [17].

III. Incomplete K-relations

Many incomplete data models do not support bag semantics, while the ones that do, treat it separately from set semantics. Our first contribution unifies both under a joint framework based on $\mathcal{K}$-relations. Concretely, we show how to define incomplete (and probabilistic) variants of a large class of $\mathcal{K}$-relations, including $\mathbb{B}$- and $\mathbb{N}$-relations. We assume a fixed set $W = \{m \mid m \in \mathbb{N} \wedge m \leq n\}$ of possible world identifiers for some $n \in \mathbb{N}$. Given the domain K of a semiring $\mathcal{K}$, we write K^W to denote the set of elements from the $|W|$-way crossproduct of K. We will annotate a tuple t with an element of K^W to store the annotation of t in each possible world.

Definition 1. *Let $\mathcal{K} = (K, \oplus_{\mathcal{K}}, \otimes_{\mathcal{K}}, \mathbb{0}_{\mathcal{K}}, \mathbb{1}_{\mathcal{K}})$ be a naturally ordered semiring. We define the possible world semiring $\mathcal{K}_W = (K^W, \oplus_{\mathcal{K}_W}, \otimes_{\mathcal{K}_W}, \mathbb{0}_{\mathcal{K}_W}, \mathbb{1}_{\mathcal{K}_W})$. The operations of this semiring are defined as follows*

$$\begin{aligned}
\forall i \in W :& \qquad \mathbb{0}_{\mathcal{K}_W}[i] = \mathbb{0}_{\mathcal{K}} \\
\forall i \in W :& \qquad \mathbb{1}_{\mathcal{K}_W}[i] = \mathbb{1}_{\mathcal{K}} \\
\forall i \in W :& \qquad (\vec{k_1} \oplus_{\mathcal{K}_W} \vec{k_2})[i] = \vec{k_1}[i] \oplus_{\mathcal{K}} \vec{k_2}[i] \\
\forall i \in W :& \qquad (\vec{k_1} \otimes_{\mathcal{K}_W} \vec{k_2})[i] = \vec{k_1}[i] \otimes_{\mathcal{K}} \vec{k_2}[i]
\end{aligned}$$

The example below illustrates the use of possible worlds semirings to encode set and bag incomplete databases.

Example 6. *Continuing Example 3, we can encode the two possible worlds of Figure 3 under bag semantics using a $\mathbb{N}^2$-relation or under set semantics using a $\mathbb{B}^2$-relation:*

id	locale	state	$\mathbb{N}^2$	$\mathbb{B}^2$
1	Lasalle	NY	**[1,1]**	**[T,T]**
2	Tucson	AZ	**[1,0]**	**[T,F]**
2	Grant Ferry	NY	**[0,1]**	**[F,T]**
3	Kingsley	NY	**[1,1]**	**[T,T]**
4	Kensington	NY	**[1,1]**	**[T,T]**

Note that the two interpretations of location 2 are recorded as two tuples which do not occur together in any possible world.

Observe that $\mathcal{K}_W$ is a semiring, since we define $\mathcal{K}_W$ using the $|W|$-way version of the *product* operation of universal algebra, and products of semirings are also semirings [7].

Possible Worlds. We can extract the $\mathcal{K}$-database for a possible world (e.g., the *best guess world*) from a $\mathcal{K}_W$-database by projecting the vector annotations down to one dimension. This can be modelled as a mapping $\text{PW}_i : K^W \to K$ where $i \in W$:

$$\text{PW}_i(\vec{k}) = \vec{k}[i] \qquad (5)$$

Recall that under possible world semantics, the result of a query Q is the set of possible worlds computed by evaluating Q over each world of the input. As a basic sanity check, we would like to ensure that query processing over $\mathcal{K}_W$-relations matches this definition. We can state possible world semantics equivalently as follows: the content of a possible world in the query result ($\text{PW}_i(Q(\mathcal{D}))$) is the result of evaluating query Q over this possible world in the input ($Q(\text{PW}_i(\mathcal{D}))$): That is, $\mathcal{K}_W$-relations have possible worlds semantics iff PW_i commutes with queries:

$$\forall i \in W : \text{PW}_i(Q(\mathcal{D})) = Q(\text{PW}_i(\mathcal{D}))$$

Recall from Section II-C that a mapping between semirings commutes with queries iff it is a semiring homomorphism. Thus, Lemma 1 shown below implies that queries over $\mathcal{K}_W$-relations have possible worlds semantics.

Lemma 1. *For any semiring $\mathcal{K}$ and possible world $i \in W$, mapping PW_i is a semiring homomorphism.*

Proof. Trivially proven by substitution of definitions.

$$\begin{aligned}
\text{PW}_i(\mathbb{0}_{\mathcal{K}_W}) &= \mathbb{0}_{\mathcal{K}_W}[i] = \mathbb{0}_{\mathcal{K}} \\
\text{PW}_i(\mathbb{1}_{\mathcal{K}_W}) &= \mathbb{1}_{\mathcal{K}_W}[i] = \mathbb{1}_{\mathcal{K}} \\
\text{PW}_i(\vec{k_1} \oplus_{\mathcal{K}_W} \vec{k_2}) &= (\vec{k_1} \oplus_{\mathcal{K}_W} \vec{k_2})[i] = \vec{k_1}[i] \oplus_{\mathcal{K}} \vec{k_2}[i] \\
&= \text{PW}_i(\vec{k_1}) \oplus_{\mathcal{K}} \text{PW}_i(\vec{k_2}) \\
\text{PW}_i(\vec{k_1} \otimes_{\mathcal{K}_W} \vec{k_2}) &= (\vec{k_1} \otimes_{\mathcal{K}_W} \vec{k_2})[i] = \vec{k_1}[i] \otimes_{\mathcal{K}} \vec{k_2}[i] \\
&= \text{PW}_i(\vec{k_1}) \otimes_{\mathcal{K}} \text{PW}_i(\vec{k_2})
\end{aligned}$$

□

Probabilistic Data. $\mathcal{K}_W$-relations admit a trivial extension to probabilistic data by defining a distribution $P : W \mapsto [0, 1]$ such that $\sum_{i \in W} P(i) = 1$. In contrast to classical frameworks for possible worlds, where the collection of worlds is a *set*, $\mathcal{K}_W$ queries preserve the same $|W|$ possible worlds[3]. Hence, the input distribution P applies, unchanged, to the $|W|$ possible query outputs.

Certain and Possible Tuples. Under set semantics, a tuple from an incomplete or probabilistic database is called *certain* if it appears in all possible worlds and *possible* if it appears in at least one possible world. Intuitively, certain (possible) is a lower (upper) bound on the content of any given possible world. We adopt this intuition to generalize these notions to $\mathcal{K}_W$-databases.

We utilize the natural order $\preceq_{\mathcal{K}}$ as introduced in Section II-C to define the greatest lower bound and least upper bound of a set of $\mathcal{K}$-elements for this purpose. For that to be well-defined we require that $\preceq_{\mathcal{K}}$ forms a lattice over K. Using the terminology from [26], we will refer to semirings which enjoy this property as *l-semirings*. A lattice over a set S and with a partial order $\leq_S$ is a structure $(S, \sqcup, \sqcap)$ where

[3]Although it has no impact on our results, it is worth noting that the worlds in a $\mathcal{K}_W$ query result may not be distinct.

$\sqcap$ (the greatest lower bound) and $\sqcup$ (the lowest upper bound) are operations over S defined for all $a, b \in S$ as:

$$a \sqcup b = \min_{\leq_S}(\{c \mid c \in S \wedge a \leq_S c \wedge b \leq_S c\})$$
$$a \sqcap b = \max_{\leq_S}(\{c \mid c \in S \wedge c \leq_S a \wedge c \leq_S b\})$$

In a lattice, $\sqcup$ and $\sqcap$ are associative, commutative, and fulfill the laws:

$$a \sqcup (a \sqcap b) = a \qquad\qquad a \sqcap (a \sqcup b) = a$$

In the following we will use $\sqcap_{\mathcal{K}}$ and $\sqcup_{\mathcal{K}}$ to denote the $\sqcap$ and $\sqcup$ operation of the lattice over $\preceq_{\mathcal{K}}$ for a semiring $\mathcal{K}$. Abusing notation, we will apply the $\sqcap_{\mathcal{K}}$ and $\sqcup_{\mathcal{K}}$ operations to elements from $\mathcal{K}_W$ with the understanding that they will be applied to the set of elements from such a vector. This is well-defined for l-semirings, since in a lattice any set of elements has a unique greatest lower bound and lowest upper bound. From here on, we will limit our discussion to l-semirings. Many semirings, including the set semiring $\mathbb{B}$ and the bag semiring $\mathbb{N}$ are l-semirings. The natural order of $\mathbb{B}$ is $F \preceq_{\mathbb{B}} T$, $k_1 \sqcup_{\mathbb{B}} k_2 = k_1 \vee k_2$, and $k_1 \sqcap_{\mathbb{B}} k_2 = k_1 \wedge k_2$. The natural order of $\mathbb{N}$ is the standard order of natural numbers, $k_1 \sqcup_{\mathbb{N}} k_2 = \max(k_1, k_2)$, and $k_1 \sqcap_{\mathbb{N}} k_2 = \min(k_1, k_2)$.

Based on $\sqcap_{\mathcal{K}}$ and $\sqcup_{\mathcal{K}}$, we define the certain and possible annotations for $\mathcal{K}_W$-elements and $\mathcal{K}_W$-databases.

$$\textsc{cert}_{\mathcal{K}}(\vec{k}) = \sqcap_{\mathcal{K}}(\vec{k}) \qquad \textsc{cert}_{\mathcal{K}}(\mathcal{D}, t) = \textsc{cert}_{\mathcal{K}}(\mathcal{D}(t))$$
$$\textsc{poss}_{\mathcal{K}}(\vec{k}) = \sqcup_{\mathcal{K}}(\vec{k}) \qquad \textsc{poss}_{\mathcal{K}}(\mathcal{D}, t) = \textsc{poss}_{\mathcal{K}}(\mathcal{D}(t))$$

As a sanity check, we observe that our definition coincides with the standard definition of certain answers for set semantics ($\mathbb{B}$), because $\textsc{cert}_{\mathbb{B}}$ corresponds to $\wedge$ (returns true if the tuple is present in all worlds). Furthermore, $\textsc{cert}_{\mathbb{N}}$ is $\min$, analogous to the definition for bag semantics given by Guagliardo and Libkin [21]. For instance, consider how to compute the certain annotation of the first tuple from Example 6: $\textsc{cert}_{\mathbb{B}}([T, T]) = T \wedge T = T$. That is, this tuple is certain. For the second tuple from this example we get $\textsc{cert}_{\mathbb{B}}([T, F]) = T \wedge F = F$. As expected, this tuple is not certain. Under bag semantics the certain multiplicity of this tuple is $\textsc{cert}_{\mathbb{N}}([1, 0]) = min(1, 0) = 0$.

IV. Uncertainty Labelings

We now formally define uncertainty labelings, which approximate certain annotations of tuples in a $\mathcal{K}_W$-database. Here, by approximate we mean that labelings may over- or under-estimate the certain annotation of a tuple with respect to the natural order of semiring $\mathcal{K}$. We use $\mathcal{K}$-relations to store labelings, i.e., the annotation of a tuple t stores its label. A labeling scheme is then a mapping which associates a $\mathcal{K}$-labeling (a $\mathcal{K}$-database) with an input $\mathcal{K}_W$-database.

Definition 2 (Uncertainty Labeling Scheme)**.** *Let $\mathcal{DB}_{\mathcal{K}}$ be the set of all $\mathcal{K}$-databases. An uncertainty labeling scheme for $\mathcal{K}_W$-database is a function $\mathsf{label}: \mathcal{DB}_{\mathcal{K}_W} \rightarrow \mathcal{DB}_{\mathcal{K}}$ such that the labeling $\mathcal{L} = \mathsf{label}(\mathcal{D})$ has the same schema as $\mathcal{D}$.*

Ideally, we would like the label (annotation) $\mathcal{L}(t)$ assigned to a tuple t by an uncertainty labeling $\mathcal{L}$ to be equal to $\textsc{cert}_{\mathcal{K}}(\mathcal{D}, t)$. However, as mentioned in the introduction, to define a tractable query semantics we will have to accept that a label $\mathcal{L}(t)$ may either overestimate or underestimate $\textsc{cert}_{\mathcal{K}}(\mathcal{D}, t)$ with respect to the natural order $\preceq_{\mathcal{K}}$. For instance, under bag semantics (semiring $\mathbb{N}$), a label n may be smaller or larger than the certain multiplicity of a tuple. A labeling is *c-sound*, i.e., does not allow for false positives, if it consistently underestimates the certain annotation of tuples, *c-complete* if it consistently overestimates certainty, and *c-correct* if it annotates every tuple with its certain annotation. We also apply this terminology to labeling schemes, e.g., a c-sound labeling scheme only produces c-sound labelings.

Definition 3 (Correctness)**.** *Let $\mathcal{L}$ be an uncertainty labeling for a $\mathcal{K}_W$-database $\mathcal{D}$.*

We call $\mathcal{L}$...	***...iff for all tuples $t \in \mathcal{D}$...***
c-sound	$\mathcal{L}(t) \preceq_{\mathcal{K}} \textsc{cert}_{\mathcal{K}}(\mathcal{D}, t)$
c-complete	$\textsc{cert}_{\mathcal{K}}(\mathcal{D}, t) \preceq_{\mathcal{K}} \mathcal{L}(t)$
c-correct	$\textsc{cert}_{\mathcal{K}}(\mathcal{D}, t) = \mathcal{L}(t)$

Observe that a labeling is both c-sound and c-complete iff it is c-correct. A query semantics over labelings should preserve the correctness properties of an input labeling if possible. That is, the result of evaluating a query Q over an uncertainty labeling $\mathcal{L}$ for a $\mathcal{K}_W$-database $\mathcal{D}$ should be a labeling for the result of evaluating the query directly over $\mathcal{D}$.

Definition 4 (Correctness Preservation)**.** *A query semantics for uncertainty labelings preserves a correctness property X (c-soundness, c-completeness, or c-correctness) wrt. a class of queries $\mathcal{C}$, if for any $\mathcal{K}_W$-database $\mathcal{D}$, labeling $\mathcal{L}$ for $\mathcal{D}$ that has property X, and query $Q \in \mathcal{C}$ we have: $Q(\mathcal{L})$ is an uncertainty labeling for $Q(\mathcal{D})$ which also has property X.*

As mentioned in the introduction (and observed elsewhere [30], [21]), false negatives are less of a concern than false positives. Thus, for cases where we cannot preserve c-correctness, we would at least opt for c-soundness. That is, we prefer labelings that underestimate the "real" $\textsc{cert}_{\mathcal{K}}(\mathcal{D}, t)$ annotation wrt. the natural order $\preceq_{\mathcal{K}}$.

V. Computing Labelings and Possible Worlds

In this section, we define efficient (`PTIME`) labeling schemes for three existing probabilistic and incomplete data models: Tuple-Independent databases [36], the disjoint-independent x-relation model from [3], and (P)C-Tables [23], [20]. We also show how to extract a best guess world from an $\mathcal{K}_W$-database derived from these models. Since computing certain answers is hard in general, our `PTIME` labeling schemes cannot be c-correct for all models. Note that our approach is applicable to a wide range of incomplete and probabilistic data models beyond the models discussed here.

A. Labeling Schemes

Tuple Independent Probabilistic Databases. A tuple independent probabilistic database (TI-DB) $\mathcal{D}$ is a database instance where each tuple t is annotated with its marginal probability $0 < P(t \in \mathcal{D}) \leq 1$. The probabilistic database represented by a TI-DB $\mathcal{D}$ is the power-set of all of its tuples: For each tuple we decide whether the tuple should be part of the possible world or not. As the name indicates, the existence of each tuple in a TI-DB is assumed to be independent of the others. Thus, the probability of a possible world D is the product of probabilities of tuples that do appear in D, and one minus the probability of tuples that do not. We define a labeling function $\mathsf{label}_{\text{TI-DB}}$ for TI-DBs that returns a $\mathbb{B}$-labeling $\mathcal{L}$ that annotates a tuple with T (certain) if its marginal probability is 1.

$$\mathcal{L}(t) = T \Leftrightarrow P(t \in \mathcal{D}) = 1$$

Theorem 1 ($\mathsf{label}_{\text{TI-DB}}$ is c-correct)**.** *Given a TI-DB $\mathcal{D}$, $\mathsf{label}_{\text{TI-DB}}(\mathcal{D})$ is a c-correct labeling.*

Proof. Trivially holds, since in any probabilistic database a tuple is certain iff $P(t) = 1$. □

C-tables. *C-Tables* [23] use a set Σ of variable symbols to define possible worlds. Tuples are annotated by a boolean expression over comparisons of values from $\Sigma \cup \mathbb{D}$, called the local condition. Each variable assignment $v : \Sigma \rightarrow \mathbb{D}$ satisfying a boolean expression called the global condition defines a possible world, derived by retaining only tuples with local conditions satisfied under v. Under the closed world assumption, C-Tables are closed under relational algebra (first order queries), and can be extended for closure under generalized projection [25] and aggregation [19]. This makes determining a c-correct labeling hard. It was shown early on that computing certain answers for first order queries is coNP-complete [38], [2] even for Codd-tables. Since we can represent the result of any first order query as a C-table and evaluating a query in this fashion is efficient, it follows that determining whether a tuple is certain in a C-table cannot be in `PTIME`. Thus, we cannot hope for an efficient c-correct labeling scheme for C-tables. Instead, consider the following sufficient, but not necessary condition for a tuple t to be certain. If (1) a tuple t in a C-table contains only constants and (2) its local condition $\phi_{\mathcal{D}}(t)$ is a tautology, then the tuple is certain. To see why this is the case, recall that under the closed-world assumption, a C-table represents a set of possible worlds, one for each valuation of the variables appearing in the C-table (to constants from $\mathbb{D}$). A tuple is part of a possible world corresponding to such a valuation if the tuple's local condition is satisfied under the valuation. Thus, a tuple consisting of constants only, with a local condition that is a tautology is part of every possible world represented by the C-table. If the local condition of a tuple is in conjunctive normal form (CNF) then checking whether it is a tautology is efficient (`PTIME`). Our labeling scheme for C-tables applies this sufficient condition and, thus, is c-sound. Formally, $\mathcal{L} = \mathsf{label}_{\text{C-TABLE}}(\mathcal{D})$, where for a C-table $\mathcal{D}$ and any tuple $t \in \textsc{TupDom}$:

$$\mathcal{L}(t) = T \Leftrightarrow \phi_{\mathcal{D}}(t) \,\text{is in CNF} \wedge (\models \phi_{\mathcal{D}}(t))$$

Theorem 2 ($\mathsf{label}_{\text{C-TABLE}}$ is c-sound)**.** *Given an incomplete database $\mathcal{D}$ encoded as C-tables, $\mathsf{label}_{\text{C-TABLE}}(\mathcal{D})$ is c-sound.*

Proof. Let $\mathcal{L} = \mathsf{label}_{\text{C-TABLE}}(\mathcal{D})$. By definition of $\mathcal{L}$, a tuple t is labeled as certain iff $\phi_{\mathcal{D}}(t)$ is in CNF and $\models \phi_{\mathcal{D}}(t)$, which means the expression $\phi_{\mathcal{D}}$ is a tautology. By definition of C-tables, a tuple t exist in a possible world if $\phi_{\mathcal{D}}(t)$ evaluates to true in that possible world. Thus, t must exist in all possible worlds if $\phi_{\mathcal{D}}(t)$ is a tautology and $\mathcal{L}$ is c-sound. □

Note that $\mathcal{L}$ is not guaranteed to be c-correct. For instance, a tuple t consisting only of constants and for which $\phi_{\mathcal{D}}(t)$ is a tautology is guaranteed to be certain, but $\mathcal{L}(t) = F$ if $\phi_{\mathcal{D}}(t)$ is not in CNF.

Example 7. *Consider a C-table consisting of two tuples $t_1 = (1, X)$ with $\phi_{\mathcal{D}}(t_1) \stackrel{def}{=} (X = 1)$ and $t_2 = (1, 1)$ with $\phi_{\mathcal{D}}(t_2) \stackrel{def}{=} (X \neq 1)$. $\mathsf{label}_{\text{C-TABLE}}$ would mark $(1, 1)$ as uncertain, because even though this tuple exists in the C-table and it's local condition is in CNF, the local condition is not a tautology. However, tuple $(1, 1)$ is certain since either $X = 1$ and then first tuple evaluates to $(1, 1)$ or $X \neq 1$ and the second tuple is included in the possible world.*

x-DBs. We use the disjoint-independent model from [3] where each relation is a set of x-tuples. Such relations are called x-relations and x-databases are sets of x-relations. An x-tuple τ is a set of tuples $\{t_1, \ldots, t_n\}$. We use $|\tau|$ to denote the number of alternatives of x-tuple τ. A probability distribution P is defined over the alternatives of each x-tuple τ, requiring that $\sum_{t \in \tau} P(t) \leq 1$. In the following we use $P(\tau)$ to denote the probability that some alternative of τ is present ($\sum_{t \in \tau} P(t)$). Each x-tuple is assumed to be independent of the others, and its alternatives are assumed to be disjoint. Thus, a possible world of an x-relation R is constructed by selecting at most one alternative $t \in \tau$ for every x-tuple τ from R, or exactly one if $P(\tau) = 1$. The probability of the resulting world is:

$$P(D) = \prod_{t \in D \wedge t \in \tau} P(t) \cdot \prod_{\nexists t \in \tau \wedge t \in D} 1 - P(\tau)$$

An incomplete version of x-relations can be defined by replacing the probability distribution with an indication whether an x-tuple is certain (one alternative has to occur in every possible world). We define a labeling scheme $\mathsf{label}_{\text{x-DB}}$ for x-relations that annotates a tuple t from an x-DB $\mathcal{D}$ with T if the x-tuple from which t stems from has a single alternative with probability 1 and F otherwise.

$$\mathcal{L}(t) = T \Leftrightarrow \exists \tau \in \mathcal{D} : t \in \tau \wedge P(t) = 1$$

Theorem 3 ($\mathsf{label}_{\text{x-DB}}$ is c-correct)**.** *Given a database $\mathcal{D}$, $\mathsf{label}_{\text{x-DB}}(\mathcal{D})$ is a c-correct labeling.*

Proof. Trivially holds, since a tuple is certain iff $P(t) = 1$. □

B. Extracting Possible Worlds

Computing some possible world is trivial for most incomplete and probabilistic data models. However, for the case of probabilistic data models we are particularly interested in the highest-probability world (the best guess world).

TI-DB. For a TI-DB $\mathcal{D}$, the best guess world consists of all tuples t such that $P(t) \geq 0.5$. To understand why this is the case recall that the probability of a world from a TI-DB is the product of the probabilities of included tuples with one minus the probability of excluded tuples. This probability is maximized by including only tuples where $P(t) \geq 0.5$.

PC-tables. For a PC-table, computing the most likely possible world reduces to answering a query over the database, which is known to be `#P` in general [36]. Specific tables (e.g., those generated by "safe" queries [36]) admit `PTIME` solutions. Alternatively, there exist a wide range of algorithms [15], [13], [14], [29] that can be used to compute an arbitrarily close approximation of the most likely world.

Disjoint-independent databases. Since the x-tuples in an x-DB are independent of each other, the probability of a possible world from an x-DB $\mathcal{D}$ is maximized by including for every x-tuple τ its alternative with the highest probability $\mathrm{argmax}_{t \in \tau} P(t)$ or no alternative if $\max_{t \in \tau} P(t) < (1 - P(\tau))$, i.e., if the probability of not including any alternative for the x-tuple is higher than the highest probability of an alternative for the x-tuple.

VI. Querying Labelings

We now study whether queries over labelings produced by *labeling schemes* such as the ones described in Section V preserve c-soundness and c-completeness. Specifically, we demonstrate that standard $\mathcal{K}$-relational query evaluation preserves *c-soundness* for $\mathcal{K}$-labeling schemes. Recall that a query semantics for labelings preserves c-soundness if for any $\mathcal{K}_W$-database $\mathcal{D}$ and a c-sound labeling $\mathcal{L}$ for $\mathcal{D}$, the output of any query Q evaluated over $\mathcal{L}$ is a c-sound labeling for $Q(\mathcal{D})$. Our result generalizes a previous result of Reiter [34] to any type of $\mathcal{K}_W$-database for which we can define an efficient c-sound labeling scheme. Furthermore, we show in Section VI-B that positive relational algebra (i.e., SPJU) queries also preserve c-completeness if the input is a labeling for a TI-DB, or if it a labeling for an x-DB and certain conditions hold on the query. We will make use of the following lemma which demonstrates that the natural order of a semiring factors through addition and multiplication. This is a known result which we only state for completeness.

Lemma 2. *Let $\mathcal{K}$ be a naturally ordered semiring. For all $k_1, k_2, k_3, k_4 \in \mathcal{K}$ we have:*

$$k_1 \preceq_{\mathcal{K}} k_3 \wedge k_2 \preceq_{\mathcal{K}} k_4 \Rightarrow k_1 \oplus_{\mathcal{K}} k_2 \preceq_{\mathcal{K}} k_3 \oplus_{\mathcal{K}} k_4$$
$$k_1 \preceq_{\mathcal{K}} k_3 \wedge k_2 \preceq_{\mathcal{K}} k_4 \Rightarrow k_1 \otimes_{\mathcal{K}} k_2 \preceq_{\mathcal{K}} k_3 \otimes_{\mathcal{K}} k_4$$

Proof. $\underline{\oplus_{\mathcal{K}}}$: Based on the definition of $\preceq_{\mathcal{K}}$, if $k \preceq_{\mathcal{K}} k'$ then there exists k'' such that $k \oplus_{\mathcal{K}} k'' = k'$. Thus, $k_3 = k_1 \oplus_{\mathcal{K}} k_1{}'$ and $k_4 = k_2 \oplus_{\mathcal{K}} k_2{}'$ for some $k_1{}'$ and $k_2{}'$. Also, $(k_1 \oplus_{\mathcal{K}} k_2) \preceq_{\mathcal{K}} (k_1 \oplus_{\mathcal{K}} k_2) \oplus_{\mathcal{K}} k''$ for any k'' and we get:

$$k_1 \oplus_{\mathcal{K}} k_2 \preceq_{\mathcal{K}} (k_1 \oplus_{\mathcal{K}} k_2) \oplus_{\mathcal{K}} (k_1{}' \oplus_{\mathcal{K}} k_2{}') = k_3 \oplus_{\mathcal{K}} k_4$$

$\underline{\otimes_{\mathcal{K}}}$: The proof for multiplication $\otimes_{\mathcal{K}}$ is similar.

$$\begin{aligned} &(k_1 \otimes_{\mathcal{K}} k_2) \\ \preceq_{\mathcal{K}} &(k_1 \otimes_{\mathcal{K}} k_2) \oplus_{\mathcal{K}} (k_1 \otimes_{\mathcal{K}} k_2{}') \oplus_{\mathcal{K}} (k_1{}' \otimes_{\mathcal{K}} k_2) \oplus_{\mathcal{K}} (k_1{}' \otimes_{\mathcal{K}} k_2{}') \\ = &(k_1 \oplus_{\mathcal{K}} k_1{}') \otimes_{\mathcal{K}} (k_2 \oplus_{\mathcal{K}} k_2{}') = k_3 \otimes_{\mathcal{K}} k_4 \end{aligned}$$ □

A. Preservation of C-Soundness

We now prove that $\mathcal{RA}^+$ over labelings preserves c-soundness. Since queries over both $\mathcal{K}_W$-databases and labelings have $\mathcal{K}$-relational query semantics, we can make use of the fact that $\mathcal{RA}^+$ over $\mathcal{K}$-relations is defined using $\oplus_{\mathcal{K}}$ and $\otimes_{\mathcal{K}}$. At a high level, the argument is as follows: (1) we show that $\mathrm{CERT}_{\mathcal{K}}$ applied to the result of an addition (or multiplication) of two $\mathcal{K}_W$-elements $\vec{k_1}$ and $\vec{k_2}$ yields a larger (wrt. $\preceq_{\mathcal{K}}$) result than adding (or multiplying) the result of applying $\mathrm{CERT}_{\mathcal{K}}$ to $\vec{k_1}$ and $\vec{k_2}$; (2) Since c-sound labelings for an input provide a lower bound on $\mathrm{CERT}_{\mathcal{K}}(\vec{k_i})$, we can apply Lemma 2 to show that the query result in the labeling is a lower bound for the result of the query over a c-correct labeling. Combining the two arguments we get c-soundness.

Functions that have the property mentioned in (1) are called superadditive and supermultiplicative. Formally, a function $f : A \to B$ where A and B are closed under addition and multiplication, and B is ordered (order $\leq_B$) is superadditive (supermultiplicative) iff for all $a_1, a_2 \in A$:

$$f(a_1 + a_2) \geq_B f(a_1) + f(a_2) \qquad \text{(superadditive)}$$
$$f(a_1 \times a_2) \geq_B f(a_1) \times f(a_2) \qquad \text{(supermultiplicative)}$$

In a nutshell, if we are given a c-correct $\mathcal{K}$-labeling, then evaluating any $\mathcal{RA}^+$-query over the labeling using $\mathcal{K}$-relational query semantics preserves c-soundness if we can prove that $\mathrm{CERT}_{\mathcal{K}}$ is superadditive and supermultiplicative.

Lemma 3. *Let $\mathcal{K}$ be a semiring. $\mathrm{CERT}_{\mathcal{K}}$ is superadditive and supermultiplicative wrt. the natural order $\preceq_{\mathcal{K}}$.*

Proof. Recall that $\oplus_{\mathcal{K}_W}$ and $\otimes_{\mathcal{K}_W}$ are defined element-wise and that $\mathrm{CERT}_{\mathcal{K}}(\vec{k}) = \sqcap_{\mathcal{K}}(\vec{k})$. Furthermore, $k_1 \preceq_{\mathcal{K}} k_2$ iff $\exists k' : k_1 \oplus_{\mathcal{K}} k' = k_2$. Consider an arbitrary $\vec{k_1}, \vec{k_2} \in K^W$. Let $k_{glb_1} = \sqcap_{\mathcal{K}}(\vec{k_1})$ and $k_{glb_2} = \sqcap_{\mathcal{K}}(\vec{k_2})$. Based on the definition of $\sqcap_{\mathcal{K}}$ this implies that for any i, $k_{glb_1} \preceq_{\mathcal{K}} \vec{k_1}[i]$ which in turn implies that $\vec{k_1}[i] = k_{glb_1} \oplus_{\mathcal{K}} k'$ for some k'. Analog, we can find a k'' such that $\vec{k_2}[i] = k_{glb_2} \oplus_{\mathcal{K}} k''$.

Superadditivity: Let $k_{glb} = \sqcap_{\mathcal{K}}(\vec{k_1} \oplus_{\mathcal{K}_W} \vec{k_2})$. We are going to prove that $k_{glb_1} \oplus_{\mathcal{K}} k_{glb_2}$ is a lower bound for $(\vec{k_1} \oplus_{\mathcal{K}_W} \vec{k_2})$, i.e., that $\forall i \in W : k_{glb_1} \oplus_{\mathcal{K}} k_{glb_2} \preceq_{\mathcal{K}} (\vec{k_1} \oplus_{\mathcal{K}_W} \vec{k_2})[i]$. Since, k_{glb} is the greatest lower bound this implies that $k_{glb_1} \oplus_{\mathcal{K}} k_{glb_2} \preceq_{\mathcal{K}} k_{glb}$. Consider an arbitrary $i \in W$. Based on the discussion above we have:

$$\begin{aligned} &(\vec{k_1} \oplus_{\mathcal{K}_W} \vec{k_2})[i] = \vec{k_1}[i] \oplus_{\mathcal{K}} \vec{k_2}[i] = k_{glb_1} \oplus_{\mathcal{K}} k' \oplus_{\mathcal{K}} k_{glb_2} \oplus_{\mathcal{K}} k'' \\ = &(k_{glb_1} \oplus_{\mathcal{K}} k_{glb_2}) \oplus_{\mathcal{K}} k' \oplus_{\mathcal{K}} k'' \succeq_{\mathcal{K}} k_{glb_1} \oplus_{\mathcal{K}} k_{glb_2} \end{aligned}$$

Thus, $k_{glb_1} \oplus_{\mathcal{K}} k_{glb_2}$ is a lower bound and since $k_{glb_1} = \text{CERT}_{\mathcal{K}}(\vec{k_1})$ and $k_{glb_2} = \text{CERT}_{\mathcal{K}}(\vec{k_2})$ it follows that $\text{CERT}_{\mathcal{K}}$ is superadditive:

$$\text{CERT}_{\mathcal{K}}(\vec{k_1}) \oplus_{\mathcal{K}} \text{CERT}_{\mathcal{K}}(\vec{k_2}) \preceq_{\mathcal{K}} \text{CERT}_{\mathcal{K}}(\vec{k_1} \oplus_{\mathcal{K}_W} \vec{k_2})$$

Supermultiplicativity: We use an analogous argument to prove supermultiplicativity. Let $k_{glb} = \text{CERT}_{\mathcal{K}}(\vec{k_1} \otimes_{\mathcal{K}} \vec{k_2})$. We will prove that $k_{glb_1} \otimes_{\mathcal{K}} k_{glb_2}$ is a lower bound for $(\vec{k_1} \otimes_{\mathcal{K}_W} \vec{k_2})$ which implies supermultiplicativity. Consider $i \in W$:

$$\begin{aligned}
&(\vec{k_1} \otimes_{\mathcal{K}_W} \vec{k_2})[i] \\
=&(k_{glb_1} \oplus_{\mathcal{K}} k') \otimes_{\mathcal{K}} (k_{glb_2} \oplus_{\mathcal{K}} k'') \\
=&(k_{glb_1} \otimes_{\mathcal{K}} k_{glb_2}) \oplus_{\mathcal{K}} (k_{glb_1} \otimes_{\mathcal{K}} k'') \oplus_{\mathcal{K}} (k' \otimes_{\mathcal{K}} k_{glb_2}) \\
&\oplus_{\mathcal{K}} (k' \otimes_{\mathcal{K}} k'') \\
\succeq_{\mathcal{K}}&(k_{glb_1} \otimes_{\mathcal{K}} k_{glb_2})
\end{aligned}$$

□

Using the superadditivity and -multiplicativity of $\text{CERT}_{\mathcal{K}}$, we now prove preservation of c-soundness. We first prove a restricted version of this result.

Lemma 4. *Let $\mathcal{D}$ be a $\mathcal{K}_W$-database and $\mathcal{L}$ be a c-correct $\mathcal{K}$-labeling for $\mathcal{D}$. $\mathcal{RA}^+$ queries over $\mathcal{L}$ preserve c-soundness.*

Proof. Consider an $\mathcal{RA}^+$ query Q and $\mathcal{D}$ an $\mathcal{K}_W$-database. To prove preservation of c-soundness, we have to show that the result of $Q(\mathcal{L})$ is a c-sound labeling for $Q(\mathcal{D})$, i.e., that for any tuple t we have $Q(\mathcal{L})(t) \preceq_{\mathcal{K}} \text{CERT}_{\mathcal{K}}(Q(\mathcal{D}), t)$. Recall that $\mathcal{RA}^+$ queries over $\mathcal{K}_W$-relations and queries over $\mathcal{K}$-labelings are defined using the semiring addition and multiplication operations. Hence, the claim

$$Q(\mathcal{L})(t) \preceq_{\mathcal{K}} \text{CERT}_{\mathcal{K}}(Q(\mathcal{D}), t)$$

follows immediately from the superadditivity and supermultiplicativity of $\text{CERT}_{\mathcal{K}}$ (Lemma 3) and the fact that $\mathcal{L}$ is a c-correct labeling. □

The major drawback of Lemma 4 is that it is limited to c-correct input labelings. Next, we show that c-soundness is still preserved even if the input labeling is only c-sound.

Theorem 4. *Let $\mathcal{D}$ be a $\mathcal{K}_W$-database, $\mathcal{L}$ a c-sound labeling for $\mathcal{D}$. $\mathcal{RA}^+$ queries over $\mathcal{L}$ preserve c-soundness.*

Proof. Since $\mathcal{L}$ is a c-sound labeling, for any tuple t we have $\mathcal{L}(t) \preceq_{\mathcal{K}} \text{CERT}_{\mathcal{K}}(\mathcal{D}, t)$. We have to prove that for any tuple t we have $Q(\mathcal{L})(t) \preceq_{\mathcal{K}} \text{CERT}_{\mathcal{K}}(Q(\mathcal{D}), t)$.

For that we show that for any $k_1, k_2 \in \mathcal{K}$ and $\vec{k_3}, \vec{k_4} \in K^W$ such that $k_1 \preceq_{\mathcal{K}} \text{CERT}_{\mathcal{K}}(\vec{k_3})$ and $k_2 \preceq_{\mathcal{K}} \text{CERT}_{\mathcal{K}}(\vec{k_4})$, we have $(k_1 \oplus_{\mathcal{K}} k_2) \preceq_{\mathcal{K}} \text{CERT}_{\mathcal{K}}(\vec{k_3} \oplus_{\mathcal{K}_W} \vec{k_4})$ and $k_1 \otimes_{\mathcal{K}} k_2 \preceq_{\mathcal{K}} \text{CERT}_{\mathcal{K}}(\vec{k_3} \otimes_{\mathcal{K}_W} \vec{k_4})$.

$$\begin{aligned}
&k_1 \oplus_{\mathcal{K}} k_2 \\
\preceq_{\mathcal{K}}&\text{CERT}_{\mathcal{K}}(\vec{k_3}) \oplus_{\mathcal{K}} \text{CERT}_{\mathcal{K}}(\vec{k_4}) && \text{(by Lemma 2)} \\
\preceq_{\mathcal{K}}&\text{CERT}_{\mathcal{K}}(\vec{k_3} \oplus_{\mathcal{K}_W} \vec{k_4}) && \text{(by Lemma 3)}
\end{aligned}$$

$$\begin{aligned}
&k_1 \otimes_{\mathcal{K}} k_2 \\
\preceq_{\mathcal{K}}&\text{CERT}_{\mathcal{K}}(\vec{k_3}) \otimes_{\mathcal{K}} \text{CERT}_{\mathcal{K}}(\vec{k_4}) && \text{(by Lemma 2)} \\
\preceq_{\mathcal{K}}&\text{CERT}_{\mathcal{K}}(\vec{k_3} \otimes_{\mathcal{K}_W} \vec{k_4}) && \text{(by Lemma 3)}
\end{aligned}$$

Since by assumption the input labeling is c-sound, we have $\mathcal{L}(t) \preceq_{\mathcal{K}} \text{CERT}_{\mathcal{K}}(\mathcal{D}, t)$ for any tuple t. Thus, based on the property we have just proven and the fact the $\mathcal{K}$-relational query semantics is defined based on the operations of semirings only, this implies that for any tuple t: $Q(\mathcal{L})(t) \preceq_{\mathcal{K}} \text{CERT}_{\mathcal{K}}(Q(\mathcal{D}), t)$. Thus, $Q(\mathcal{L})$ is a c-sound labeling for $Q(\mathcal{D})$. □

B. *Preservation of C-Completeness*

TI-DBs. We now demonstrate that positive queries preserve c-completeness if the input is a $\mathcal{K}$-labeling produced by our labeling scheme for TI-DBs. Recall that we had already proven that the labeling scheme for TI-DBs presented in Section V is c-complete. The proof of preservation of c-completeness is based on the following observation: If two $\mathcal{K}_W$-elements $\vec{k_1}$ and $\vec{k_2}$ are minimal in the same possible world, i.e., there exists a possible world i such that $\sqcap_{\mathcal{K}}(\vec{k_1}) = \vec{k_1}[i]$ and $\sqcap_{\mathcal{K}}(\vec{k_2}) = \vec{k_2}[i]$, then $\sqcap_{\mathcal{K}}$ commutes with addition and multiplication. Thus, standard $\mathcal{K}$-relational query evaluation semantics over c-complete labelings preserves c-completeness if the aforementioned property holds. We then prove that every $\mathcal{K}_W$ encoding of a TI-DB database has this property.

Lemma 5. *Let $\vec{k_1}, \vec{k_2} \in K^W$ for some possible world semiring $\mathcal{K}_W$. If there exists $i \in W$ such that $\sqcap_{\mathcal{K}}(\vec{k_1}) = \vec{k_1}[i]$ and $\sqcap_{\mathcal{K}}(\vec{k_2}) = \vec{k_2}[i]$, then the following holds:*

$$\sqcap_{\mathcal{K}}(\vec{k_1} \oplus_{\mathcal{K}_W} \vec{k_2}) = \sqcap_{\mathcal{K}}(\vec{k_1}) \oplus_{\mathcal{K}} \sqcap_{\mathcal{K}}(\vec{k_2}) = (\vec{k_1} \oplus_{\mathcal{K}_W} \vec{k_2})[i]$$
$$\sqcap_{\mathcal{K}}(\vec{k_1} \otimes_{\mathcal{K}_W} \vec{k_2}) = \sqcap_{\mathcal{K}}(\vec{k_1}) \otimes_{\mathcal{K}} \sqcap_{\mathcal{K}}(\vec{k_2}) = (\vec{k_1} \otimes_{\mathcal{K}_W} \vec{k_2})[i]$$

Proof. Recall that PW_i is a homomorphism (Lemma 1), so $(\vec{k_1} \oplus_{\mathcal{K}_W} \vec{k_2})[i] = \vec{k_1}[i] \oplus_{\mathcal{K}} \vec{k_2}[i]$ and $\sqcap_{\mathcal{K}}(\vec{k_j}) = \vec{k_j}[i]$ for $j \in \{1, 2\}$. Thus, $(\vec{k_1} \oplus_{\mathcal{K}_W} \vec{k_2})[i] = \sqcap_{\mathcal{K}}(\vec{k_1}) \oplus_{\mathcal{K}} \sqcap_{\mathcal{K}}(\vec{k_2})$. It remains to be shown that $\sqcap_{\mathcal{K}}(\vec{k_1} \oplus_{\mathcal{K}_W} \vec{k_2}) = (\vec{k_1} \oplus_{\mathcal{K}_W} \vec{k_2})[i]$ which holds if for any $j \neq i \in W$ we have $(\vec{k_1} \oplus_{\mathcal{K}_W} \vec{k_2})[i] \preceq_{\mathcal{K}} (\vec{k_1} \oplus_{\mathcal{K}_W} \vec{k_2})[j]$. Since $\vec{k_1}[i] = \sqcap_{\mathcal{K}}(\vec{k_1})$ and $\sqcap_{\mathcal{K}}$ is defined based on the natural order, we know that $\vec{k_1}[i] \preceq_{\mathcal{K}} \vec{k_1}[j]$ and analog for $\vec{k_2}$ we have $\vec{k_2}[i] \preceq_{\mathcal{K}} \vec{k_2}[j]$. Based on Lemma 2 this implies $(\vec{k_1} \oplus_{\mathcal{K}_W} \vec{k_2})[i] \preceq_{\mathcal{K}} (\vec{k_1} \oplus_{\mathcal{K}_W} \vec{k_2})[j]$. The proof for multiplication is analog using Lemma 2 to show that $(\vec{k_1} \otimes_{\mathcal{K}_W} \vec{k_2})[i] \preceq_{\mathcal{K}} (\vec{k_1} \otimes_{\mathcal{K}_W} \vec{k_2})[j]$ for any $j \in W$. □

To demonstrate c-completeness preservation for TI-DBs we have to demonstrate that the encoding of a TI-DB as a $\mathcal{K}_W$-database fulfills the precondition of Lemma 5.

Lemma 6. *Let $\mathcal{D}$ be a $\mathcal{K}_W$-database that represents a TI-DB. Then there exists $i \in W$ such that for any tuple t:*

$$\sqcap_{\mathcal{K}}(\mathcal{D}(t)) = \mathcal{D}(t)[i].$$

Proof. Consider the possible world D defined as follows:

$$D(t) = \begin{cases} \sqcap_{\mathcal{K}}(\mathcal{D}(t)) & \text{if } P(t) = 1 \\ \mathbb{0}_{\mathcal{K}} & \text{otherwise} \end{cases}$$

This world exists, because in a TI-DB all tuples with probability equals to 1 carry the same annotation in every possible world i. Furthermore, since the tuples are assumed to be independent events there exists a possible world that does contain none of the tuples with probability less than one. Let i denote the identifier of D in $\mathcal{D}$, i.e., $\text{PW}_i(\mathcal{D}) = D$. We claim that $\sqcap_{\mathcal{K}}(\mathcal{D}(t)) = D(t) = \mathcal{D}(t)[i]$ for all $t \in \text{TUPDOM}$. We have to distinguish two cases. Either $P(t) = 1$ and $\mathcal{D}(t)[i] = \mathcal{D}(t)[j]$ for any $j \in W$ and, thus, $\sqcap_{\mathcal{K}}(\mathcal{D}(t)) = \mathcal{D}(t)[i]$. Otherwise, $P(t) < 1$ and $D(t) = \mathcal{D}(t)[i] = \mathbb{0}_{\mathcal{K}}$. Since, $\mathbb{0}_{\mathcal{K}}$ is the smallest element in $\mathcal{K}$ wrt. $\preceq_{\mathcal{K}}$ from this follows that $\sqcap_{\mathcal{K}}(\mathcal{D}(t)) = \mathbb{0}_{\mathcal{K}} = \mathcal{D}(t)[i]$. □

Lemmas 5 and 6 together imply that our labeling approach preserves c-completeness if the input is a TI-DB.

Corollary 1. *Let $\mathcal{L}$ be a labeling for a TI-DB $\mathcal{D}$ computed as $\mathsf{label}_{TI}(\mathcal{D})$. Then $\mathcal{RA}^+$ over $\mathcal{L}$ preserves c-completeness.*

x-DBs. In general, $\mathcal{RA}^+$ queries over labelings derived from x-DBs do not preserve c-completeness. However, not all is lost. We present a sufficient condition for a query to preserve c-completeness over such a labeling. To this end, we define *x-keys*, constraints that ensure that alternatives within the scope of an x-tuple are not all identical if projected on a set of attributes A. Since our labeling scheme for x-DBs is c-complete, queries preserve c-completeness unless a result tuple that is certain is derived from multiple *correlated* uncertain input tuples. Since x-tuples from an x-DB are independent of each other, this can only be the case if a result tuple is derived from alternatives of an x-tuple τ from every possible world (i.e., where $P(\tau) = 1$). Such a situation can be avoided if it is guaranteed that it is impossible for a result tuple to be derived from all alternatives of an x-tuple. Intuitively, if we ensure that two different alternatives of an x-tuple are only used to generate different outputs then this is guaranteed to hold.

Definition 5 (x-key)**.** *Let R be an x-relation with schema $\mathbf{R}$. A set of attributes $A \subseteq \mathbf{R}$ is called an x-key for R iff*

$$\forall \tau \in R : (\exists t_1, t_2 \in \tau : t_1[A] \neq t_2[A]) \vee P(\tau) < 1 \vee |\tau| = 1$$

Intuitively, an x-key is a set of attributes A such that for any x-tuple τ with $P(\tau) = 1$ and more than one alternative, there exists at least two alternatives that differ in A. The following lemma states that a superset of an x-key is also an x-key.

Lemma 7. *Let $A \subseteq B \subseteq \mathbf{R}$ where $\mathbf{R}$ is the schema of an x-relation R. If A is an x-key for R, then so is B.*

Proof. Whether any of the last two conditions of Definition 5 holds for an x-tuple is independent of the particular choice of x-key. If the first condition is true, then two alternatives of the x-tuple differ on A which trivially implies that they differ on a superset of A. □

We now prove that for any x-relation R, if a conjunctive, self-join free query Q (a query using only selection, projection, and crossproduct that accesses no relation more than once) returns at least one x-key per accessed input relation, then the query preserves c-completeness.

Theorem 5. *Let $\mathcal{L}$ be a labeling for an x-DB $\mathcal{D}$ computed using $\mathsf{label}_{\text{x-DB}}$. Consider a conjunctive query Q in canonical form $\pi_{\mathcal{A}}(\sigma_\theta(R_1 \times \ldots \times R_n))$ with $R_i \neq R_j$ for all $i \neq j \in \{1, \ldots, n\}$. Query Q preserves c-completeness if $\mathcal{A}$ contains an x-key for every relation R_i accessed by Q.*

Proof. Let $\mathcal{D} = \{R_1, \ldots, R_n\}$ be an x-database, $\mathcal{D}' = \{R'_1, \ldots, R'_n\}$ its encoding as a $\mathbb{B}_W$-database, $\mathcal{L}$ a c-complete labeling for $\mathcal{D}'$ derived using $\mathsf{label}_{\text{x-DB}}$, and Q be a selfjoin-free query of the form $\pi_{\mathcal{A}}(\sigma_\theta(R_1 \times \ldots \times R_n))$ such that $\mathcal{A}$ contains an x-key for every relation R_i for $i \in \{1, \ldots, n\}$. Note that any selfjoin-free $\mathcal{RA}^+$ query without union can be brought into this form. We have to show that $Q(\mathcal{L})$ is a c-complete labeling for $Q(\mathcal{D}')$. We prove this claim by contradiction. For sake of the contradiction assume that $Q(\mathcal{L})$ is not a c-complete labeling. Then there has to exist a result tuple $t \in Q(\mathcal{D}')$ such that $Q(\mathcal{L})(t) = F$ and $\text{CERT}_{\mathbb{B}}(Q(\mathcal{D}')(t)) = T$. Recall that $\oplus_{\mathbb{B}} = \vee$ and $\otimes_{\mathbb{B}} = \wedge$. Unfolding definitions of relational algebra operators over $\mathcal{K}$-relations we get:

$$Q(\mathcal{L})(t) = \bigvee_{u:u[A]=t \wedge \forall i \in \{1,\ldots,n\}: u[R_i]=t_i} \left(\bigwedge_{i=1}^{n} \mathcal{L}(t_i) \right) \wedge \theta(u)$$

$$Q(\mathcal{D}')(t) = \sum_{u:u[A]=t \wedge \forall i \in \{1,\ldots,n\}: u[R_i]=t_i} R'_1(t_1) \otimes_{\mathbb{B}_W} \ldots \otimes_{\mathbb{B}_W} R'_n(t_n) \otimes_{\mathbb{B}_W} \theta(u)$$

Note that for result tuples u of the crossproduct for which $u \not\models \theta$ we have $\theta(u) = F$ (respective $\theta(u) = \mathbb{0}_{\mathbb{B}_W}$). Thus, any monomial (product) corresponding to such a u will evaluate to F ($\mathbb{0}_{\mathbb{B}_W}$). Thus, we can equivalently express the above expressions as shown below where the j values identify monomials for which $u \models \theta$ WLOG assuming that there are m such monomials for some $m \in \mathbb{N}$.

$$Q(\mathcal{L})(t) = \bigvee_{\forall j \in \{1,\ldots,m\}} \left(\bigwedge_{i=1}^{n} \mathcal{L}(t_{j_i}) \right)$$

$$Q(\mathcal{D}')(t) = \sum_{\forall j \in \{1,\ldots,m\}} \prod_{i=1}^{n} R'_i(t_{j_i})$$

In the following we will use b_{j_i} to denote $\mathcal{L}(t_{j_i})$ and $\vec{k_{j_i}}$ to denote $R'_i(t_{j_i})$. Based on the assumption we have made we know that

$$\bigvee_{\forall j \in \{1,\ldots,m\}} \left(\bigwedge_{i=1}^{n} b_{j_i} \right) = F$$

So this can only be the case if for every $j \in \{1, \ldots, m\}$ there exists $f \in \{1, \ldots, n\}$ such that $b_{j_f} = F$. For any $j \in \{1, \ldots, m\}$ let min_j denote the smallest such f, i.e., the first element in the j^{th} conjunct that is false and let t_{min_j} denote the corresponding tuple. Based on the fact that $\mathcal{L} = \mathsf{label}_{\text{X-DB}}(\mathcal{D}')$ we know that if $\mathcal{L}(t_{min_j}) = F$ then $P(t_{min_j}) < 1$. We will use this fact to derive a contradiction with the assumption $\textsc{cert}_{\mathbb{B}}(Q(\mathcal{D}')(t)) = T$. For that, we partition the set of monomials from $Q(\mathcal{D}')(t)$ into two subsets M_1 and ${M_1}^{\mathcal{C}}$ where M_1 contains the identifiers j of all monomials such that $min_j = 1$ and ${M_1}^{\mathcal{C}}$ contains all remaining monomials. We will show that $\textsc{cert}_{\mathbb{B}}(\sum_{j \in M_1} \prod_{i=1}^{n} k_{j_i}) = F$, then $\textsc{cert}_{\mathbb{B}}(\sum_{j \in {M_1}^{\mathcal{C}}} \prod_{i=1}^{n} k_{j_i}) = F$, and finally $\textsc{cert}_{\mathbb{B}}(Q(\mathcal{D}')(t)) = \textsc{cert}_{\mathbb{B}}(\sum_{j \in M_1} \prod_{i=1}^{n} k_{j_i} \oplus_{\mathbb{B}_W} \sum_{j \in {M_1}^{\mathcal{C}}} \prod_{i=1}^{n} k_{j_i}) = F$ which is the contradiction we wanted to derive.

First, consider $\sum_{j \in M_1} \prod_{i=1}^{n} k_{j_i}$. Since $\otimes_{\mathbb{B}} = \wedge$ and $\otimes_{\mathbb{B}_W}$ is defined as point-wise application of $\wedge$ to a vector $\vec{k} \in \mathbb{B}_W$ we have $\vec{k} \otimes_{\mathbb{B}_W} \vec{k'} \preceq_{\mathbb{B}_W} \vec{k}$ for any $\vec{k}$ and $\vec{k'}$. Thus, $\sum_{j \in M_1} \prod_{i=1}^{n} k_{j_i} \preceq_{\mathbb{B}_W} \sum_{j \in M_1} k_{j_1}$. We will show that $\textsc{cert}_{\mathbb{B}}(\sum_{j \in M_1} k_{j_1}) = F$ from which follows $\textsc{cert}_{\mathbb{B}}(\sum_{j \in M_1} \prod_{i=1}^{n} k_{j_i}) = F$.

By construction we have $P(t_{j_1}) < 1$ for all j in M_1. Now consider the set of x-tuples from R_1 for which the tuples t_{j_1} are alternatives. WLOG let $\tau_1, \ldots, \tau_l$ be these x-tuples. Now consider an arbitrary x-tuple τ from this set and let $s_1, \ldots, s_o$ be its alternatives that are present in M_1. We know that $P(s_i) < 1$ for all s_i. We distinguish 2 cases: either $P(\tau) < 1$ or $P(\tau) = 1$. In the latter case based on the fact that $\mathcal{A}$ contains an x-key for R_1 we know that there exists at least one alterative s of τ that is neither in M_1 nor in ${M_1}^{\mathcal{C}}$. To see why this is the case observe that its presence in M_1 would violate the x-key while by construction ${M_1}^{\mathcal{C}}$ only contains tuples t from R'_1 with $P(t) = 1$. Next we construct a possible world $w \in W$ from $\mathcal{D}$ which does not contain any of the t_{j_1} which means that $k_{j_1}[w] = F$. In turn, this implies that $\sum_{j \in M_1} k_{j_1} = F$. We construct w as follows: for every x-tuple τ from $\tau_1, \ldots, \tau_l$ we either include no alternative of τ if $P(\tau) < 1$ or an alternative that is not present in M_1. Now further partition ${M_1}^{\mathcal{C}}$ into two subsets: M_2 which contains all monomials for which $min_j = 2$ and ${M_2}^{\mathcal{C}}$ for all remaining monomials. Then using an argument symmetric to the one given for M_1 above we can construct a possible world for which $\sum_{j \in M_2} \prod_{i=1}^{n} t_{j_i}[w] = F$ and, thus, $\textsc{cert}_{\mathbb{B}}(\sum_{j \in M_2} \prod_{i=1}^{n} k_{j_i}) = F$. Because the x-tuples from M_1 and M_2 are from different relations there is no overlap between these sets of x-tuples. Based on the independence of x-tuples in x-DBs this implies that we can also construct a possible world w where $\sum_{j \in M_1} \prod_{i=1}^{n} k_{j_i}[w] \oplus_{\mathbb{B}_W} \sum_{j \in M_2} \prod_{i=1}^{n} k_{j_i}[w] = F$ and, thus, $\textsc{cert}_{\mathbb{B}}(\sum_{j \in M_1} \prod_{i=1}^{n} k_{j_i} \oplus_{\mathbb{B}_W} \sum_{j \in M_2} \prod_{i=1}^{n} k_{j_i}) = F$. We can now continue this construction to include M_3, M_4, and so on. Note that we are guaranteed that M_n contains all monomials that will be left over at this point, because we started from the observation that at least one k in every monomial corresponds to a tuple t which has $P(t) < 1$. It follows that

$$\textsc{cert}_{\mathbb{B}}(Q(\mathcal{D}')(t)) = \textsc{cert}_{\mathbb{B}} \left(\sum_{o=1}^{n} \left(\sum_{j \in M_o} \prod_{i=1}^{n} k_{j_i} \right) \right) = F$$

which contradicts our assumption that $\textsc{cert}_{\mathbb{B}}(Q(\mathcal{D}')(t)) = T$ and thus concludes the proof. □

VII. UA-Databases

We now introduce *UA-DBs* (uncertainty-annotated databases) which are lossless encodings of both (1) a single possible world D from a $\mathcal{K}_W$-database $\mathcal{D}$ and (2) an uncertainty labeling $\mathcal{L}$ for $\mathcal{D}$. Importantly, we can efficiently restore D and $\mathcal{L}$ from a UA-DB. This is achieved by annotating every tuple with a pair $[d, c] \in K^2$ where d records the tuple's annotation in the possible world and c stores the tuple's annotation in the uncertainty labeling. In other words, if a tuple t is annotated with $[d, c]$, then $\mathcal{L}(t) = c$ and $\textsc{pw}_i(\mathcal{D})(t) = d$ if i is the identifier of the encoded possible world. As motivated in the introduction, by annotating a possible world with uncertainty information we get the best of certain answer semantics and best guess query processing. Since every possible world by definition is a superset of the certain tuples, a UA-DB contains all certain answers, even though its labeling may misclassify some of them as uncertain. Next, we define a UA-semiring as a $\mathcal{K}^2$-semiring, i.e., the direct product of a semiring $\mathcal{K}$ with itself (Section VII-A). Afterwards, we prove that the result of a query over a UA-DB encodes the result of the query over the input possible world and uncertainty labeling.

A. UA-semirings

In the following we will write kk' instead of $k \otimes_{\mathcal{K}} k'$ if the semiring $\mathcal{K}$ is clear from the context. Recall that $\mathcal{K}^2$ denotes the direct product semiring $\mathcal{K}^2 = (K^2, \oplus_{\mathcal{K}^2}, \otimes_{\mathcal{K}^2}, \mathbb{0}_{\mathcal{K}^2}, \mathbb{1}_{\mathcal{K}^2})$ where the operations are defined pointwise, e.g., $[k_1, {k_1}'] \otimes_{\mathcal{K}^2} [k_2, {k_2}'] = [k_1 \otimes_{\mathcal{K}} k_2, {k_1}' \otimes_{\mathcal{K}} {k_2}']$.

Definition 6 (UA-semiring)**.** *Let $\mathcal{K}$ be a semiring. We define the corresponding UA-semiring*

$$\mathcal{K}_{UA} = \mathcal{K}^2$$

Note that if $\mathcal{K}$ is a commutative semiring, then $\mathcal{K}_{UA}$ is a commutative semiring, because, as mentioned earlier, the product of two semirings is also a semiring.

B. Creating UA-DBs

We now discuss how to derive UA-relations from a $\mathcal{K}_W$-database. Consider a $\mathcal{K}_W$-database $\mathcal{D}$, let D be one of its possible worlds and $\mathcal{L}$ an uncertainty labeling for $\mathcal{D}$. We can compute the possible world D and labeling $\mathcal{L}$ using the techniques presented in Section V. We construct a UA-DB D_{UA} as an *encoding* of D and $\mathcal{L}$ by setting for every tuple t:

$$D_{UA}(t) = [D(t), \mathcal{L}(t)]$$

Given a UA-DB D_{UA} derived from a possible world D, we would like to be able to restore D from D_{UA}. Similarly, we would like to be able to recover the labeling $\mathcal{L}$ we started from. If this is possible then we can confidently claim that D_{UA} encodes the possible world D and labeling $\mathcal{L}$. For that we define two morphisms $\mathcal{K}^2 \to \mathcal{K}$ which for any $[d, c] \in K^2$ are defined as:

$$h_{cert}([d, c]) = c \qquad h_{det}([d, c]) = d$$

Note that by construction if an UA-DB D_{UA} is an *encoding* of a possible world D and a labeling $\mathcal{L}$ of a $\mathcal{K}_W$-database $\mathcal{D}$ then: $h_{det}(D_{UA}) = D$ **and** $h_{cert}(D_{UA}) = \mathcal{L}$

C. Preservation of C-soundness and Encoded Worlds

We now prove that query evaluation over UA-DBs that are encodings of a possible world D and labeling $\mathcal{L}$ preserves c-soundness for the labeling $\mathcal{L}$ and the possible world D.

Theorem 6 (Preservation of Encoded Worlds)**.** *Let D_{UA} be a UA-database that is an encoding for a possible world D from a $\mathcal{K}_W$-database $\mathcal{D}$. Then $Q(D_{UA})$ is an encoding for $Q(D)$, i.e., $h_{det}(Q(D_{UA})) = Q(h_{det}(D_{UA}))$.*

Proof. Note that $h_{det}(Q(D_{UA})) = Q(h_{det}(D_{UA}))$ holds if h_{det} is a homomorphism. Thus, it suffices to show that h_{det} is a homomorphism. Let $k_1 = [d_1, c_1]$ and $k_2 = [d_2, c_2]$.

$$h_{det}(\mathbb{0}_{\mathcal{K}_{UA}}) = \mathbb{0}_{\mathcal{K}}$$
$$h_{det}(\mathbb{1}_{\mathcal{K}_{UA}}) = \mathbb{1}_{\mathcal{K}}$$

$$\begin{aligned} h_{det}(k_1 \oplus_{\mathcal{K}_{UA}} k_2) &= h_{det}([d_1 \oplus_{\mathcal{K}} d_2, c_1 \oplus_{\mathcal{K}} c_2]) \\ &= d_1 \oplus_{\mathcal{K}} d_2 \\ &= h_{det}(k_1) \oplus_{\mathcal{K}} h_{det}(k_2) \end{aligned}$$

$$\begin{aligned} h_{det}(k_1 \otimes_{\mathcal{K}_{UA}} k_2) &= h_{det}([d_1 \otimes_{\mathcal{K}} d_2, c_1 \otimes_{\mathcal{K}} c_2]) \\ &= d_1 \otimes_{\mathcal{K}} d_2 \\ &= h_{det}(k_1) \otimes_{\mathcal{K}} h_{det}(k_2) \end{aligned}$$

□

Recall that queries over labelings preserves c-soundness.

Theorem 7 (Preservation of C-soundness)**.** *Let D_{UA} be a UA-database that is a c-sound labeling for a $\mathcal{K}_W$-database $\mathcal{D}$. $\mathcal{RA}^+$ queries over D_{UA} preserve c-soundness.*

Proof. Mapping $h_{cert}(D_{UA})$ derives a labeling from a D_{UA} by extracting the uncertainty labeling that was encoded in the second component of tuple annotations in D_{UA}. Note that h_{cert} is a homomorphism from $\mathcal{K}_{UA}$ to $\mathcal{K}$ and, thus, computes with queries. Now we have $h_{cert}(Q(D_{UA})) = Q(h_{cert}(D_{UA}))$, i.e., the uncertainty labeling generated by evaluating Q over D_{UA} is equivalent to the result of evaluating the query over the labeling corresponding to D_{UA}. Recall that we have already demonstrated that $\mathcal{RA}^+$ queries over labelings preserve c-soundness. Hence, $\mathcal{RA}^+$ over UA-DBs preserves c-soundness. □

$[\![R]\!]_{UA}$	$=$	A Labeled `R` (see Section V)
$[\![\sigma_\theta(Q)]\!]_{UA}$	$=$	**SELECT** `*` **FROM** `Q` **WHERE** θ
$[\![\pi_{A_1 \dots A_n}(Q)]\!]_{UA}$	$=$	**SELECT** `A1, ..., AN, C` **FROM** `Q`
$[\![Q_1 \bowtie_\theta Q_2]\!]_{UA}$	$=$	**SELECT** `Q1.*, Q2.*,` `Q1.C * Q2.C` **AS** `C` **FROM** `Q1, Q2` **WHERE** θ
$[\![Q_1 \cup Q_2]\!]_{UA}$	$=$	`Q1` **UNION ALL** `Q2`

Fig. 6: Query rewrite rules

VIII. Implementation

We now discuss the implementation of a UA-DB as a query rewriting front-end built on top of a relational DBMS. A $\mathcal{K}_{UA}$-relation with schema $\mathbf{R}(A_1, \dots, A_n)$ annotated with a pair of $\mathcal{K}$-elements $[d, c]$ is encoded by a $\mathcal{K}$-relation $\mathbf{R}'(A_1, \dots, A_n, C)$ where the annotation of a tuple encodes d and attribute C stores c. We specifically implement UA-DBs for bag semantics, as this is the model used by most DBMS. Note that in contrast to $\mathbb{N}$-relations where the multiplicity of a tuple is stored as its annotation, relational databases represent a tuple t with multiplicity n as n copies of t. While in principle we could use attribute C to store c for each copy of t, alternatively we can use C as a boolean marker and mark c copies of t as certain (1) and the remaining $d - c$ copies as uncertain (0) as shown in the example in Section I. We believe that this approach is easier to interpret for users and since it does not have any drawbacks apply it in our implementation.

Our frontend rewriting engine receives queries of the form $Q(D_{UA})$ over a $\mathbb{N}_{UA}$-annotated database D_{UA} with schema $\{ \mathbf{R}_i(A_1, \dots, A_n) \}$. It rewrites such a query into an equivalent query $[\![Q]\!]_{UA}(D)$ over a classical bag-relational database D with schema $\{ \mathbf{R}'_i(A_1, \dots, A_n, C) \}$ where $C \in \{0, 1\}$ denotes the uncertainty label. The rewriting $[\![\cdot]\!]_{UA}$ is defined through a set of rules given in Figure 6. To support queries over a wide range of incomplete data models we allow the user to specify the data model of each input relation. Our rewriting engine uses SQL implementations of the labeling schemes and extraction of best guess worlds from Section V to transform such an input relation into our encoding of $\mathbb{N}_{UA}$-relations. We implement our approach as a middleware over a database system through an extension of SQL. An input query is first parsed, translated into a relational algebra graph, rewritten using $[\![\cdot]\!]_{UA}$, and then converted back to SQL for execution.

A. Relational Algebra Rewriting and Correctness

To prove that the rewriting defined above is correct, we first formally define the function Enc implementing the encoding of a $\mathbb{N}_{UA}$-database as a $\mathbb{N}$-database and restate $[\![\quad]\!]_{UA}$ as relational algebra rewriting rules. Afterwards, we prove that this rewriting correctly encodes $\mathbb{N}_{UA}$ query semantics. In the following, we use $\{t \mapsto k\}$ to denote a singleton relation where tuple t is annotated with k and all other tuples are annotated with 0. Recall that $arity(R)$ denotes the arity (number of attibutes) of a relation.

Definition 7 (Multiset encoding)**.** $\textsc{Enc}(R)$ *is a function from* $\mathbb{N}_{UA}$*-relations to* $\mathbb{N}$*-relations. Let* R *be a* $\mathbb{N}_{UA}$*-relation with schema* $A_1, \ldots, A_n$*. Let* R' *be an* $\mathbb{N}$*-relation with schema* $A_1, \ldots, A_n, U$ *that is the result of* $\textsc{Enc}(R)$ *for some* R*.* $\textsc{Enc}$ *and its inverse are defined as:*

$$\textsc{Enc}(R) = \bigcup_{t \in \mathbb{D}^{arity(R)}} \{(t,1) \mapsto h_{cert}(R(t))\} \cup \{(t,0) \mapsto h_{det}(R(t)) - h_{cert}(R(t))\}$$

$$\textsc{Enc}^{-1}(R') = \bigcup_{t \in \mathbb{D}^{arity(R)}} t \mapsto (R'(t,0) + R'(t,1), R'(t,1))$$

We define $\textsc{Enc}$ over databases as applying $\textsc{Enc}$ to every relation in the database. Note that even though we define the encoding for bag semantics here, it can be generalized to any $\mathcal{K}_{UA}$ where semiring $\mathcal{K}$ has a monus [17] by replacing $-$ with $\ominus_{\mathcal{K}}$ (the monus operation). Next, we define the relational algebra version of our rewriting $[\![\cdot]\!]_{UA}$ that translates an input query into a query over the encoding produced by $\textsc{Enc}$. Again, the rewriting is defined through a set of rules (one per relational algebra operator). The rules are shown in Figure 7. Here $\mathbf{D}(Q)$ denotes the schema of the result of query Q and $e \rightarrow a$ used in generalized projection expressions denotes projecting on the result of evaluating expression e and calling the resulting attribute a.

Theorem 8. *Let* D_{UA} *be a* $\mathbb{N}_{UA}$*-database and* Q *an* $\mathcal{RA}^+$ *query. The following holds:*

$$Q(D_{UA}) = \textsc{Enc}^{-1}([\![Q]\!]_{UA}(\textsc{Enc}(D_{UA})))$$

Proof. Straightforward induction over the structure of queries.

Base case: $Q = R$: WLOG consider a tuple t and let $R(t) = [d, c]$. We know that

$$\textsc{Enc}(R)(t,0) = h_{det}(R(t)) - h_{cert}(R(t)) = d - c$$

and

$$\textsc{Enc}(R)(t,1) = h_{cert}(R(t)) = c$$

Let $R' = \textsc{Enc}^{-1}(Q_{[\![ ]\!]_{UA}}(\textsc{Enc}(R))) = \textsc{Enc}^{-1}(\textsc{Enc}(R))$. Then $R'(t) = [R(t,0) + R(t,1), R(t,1)] = [d - c + c, c] = [d, c]$.

Induction Step: Assume that the claim holds for queries Q_1 and Q_2, we have to show that it also holds for applying an operator of $\mathcal{RA}^+$ to the result of these queries. We use $Q_{[\![ ]\!]_{UA}i} = [\![Q_i]\!]_{UA}$ and $R_i = Q_i(D_{UA})$.

Selection $\sigma_\theta(Q_1)$: Note that

$$[\![\sigma_\theta(Q_1)]\!]_{UA} = \sigma_\theta([\![Q_1]\!]_{UA})$$

Consider a tuple t with $R_1(t) = [d, c]$. Let ${R_1}' = \textsc{Enc}^{-1}(\sigma_\theta(\textsc{Enc}(R_1)))$. We have $\sigma_\theta(R_1)(t) = R_1(t) \otimes_{\mathbb{N}_{UA}} \theta(t)$ and

$$\sigma_\theta(\textsc{Enc}(R_1)(t,0)) = (d - c) \cdot \theta(t,0)$$

$$\sigma_\theta(\textsc{Enc}(R_1)(t,1)) = c \cdot \theta(t,1)$$

Since the selection condition does not access attribute U, we have

$$\theta(t,0) = \theta(t,1) = 1 \Leftrightarrow \theta(t) = [0,1]$$

Applying the definition of $\textsc{Enc}^{-1}$, we get

$${R_1}'(t) = [(d - c) \cdot \theta(t,0) + c \cdot \theta(t,1), c \cdot \theta(t,1)]$$

We now distinguish two cases: either $t \models \theta$ and $t \not\models \theta$. First consider the case where $t \models \theta$. Then, $\theta(t,0) = \theta(t,1) = 1$ and we get

$${R_1}'(t) = [(d - c + c) \cdot 1, c \cdot 1] = [d, c] = R_1(t) = R_1(t) \otimes_{\mathbb{N}_{UA}} \theta(t)$$

Now consider the case $t \not\models \theta$. Then, $\theta(t,0) = \theta(t,1) = 0$ and we get

$${R_1}'(t) = [(d - c + c) \cdot 0, c \cdot 0] = [0, 0] = R_1(t) \otimes_{\mathbb{N}_{UA}} \theta(t)$$

Natural Join $Q_1 \bowtie_\theta Q_2$: Let

$$R' = \textsc{Enc}^{-1}([\![Q_1 \bowtie_\theta Q_2]\!]_{UA}(\textsc{Enc}(R_1), \textsc{Enc}(R_2))$$

and consider a tuple t with $t \models \theta$ and let $t_1 = t[R_1]$, $t_2 = t[R_2]$, $R_i(t_i) = [d_i, c_i]$ for $i \in \{1, 2\}$, and $Q_{res} = [\![Q_1 \bowtie_\theta Q_2]\!]_{UA}$.

$$Q_{res} = \pi_{\textsc{Sch}(Q_1 \bowtie_\theta Q_2), min(Q_1.C, Q_2.C) \rightarrow C}(Q_{join})$$
$$Q_{join} = [\![Q_1]\!]_{UA} \bowtie_\theta [\![Q_2]\!]_{UA}$$

Based on the induction assumption we have $[\![Q_i]\!]_{UA}(t_i, 0) = d_i - c_i$ and $[\![Q_i]\!]_{UA}(t_i, 1) = c_i$. Mapping $\textsc{Enc}$ creates two version of t_i, thus, there are 4 ways of joining these versions:

$$\begin{aligned} Q_{join}(t,0,0) &= [\![Q_1]\!]_{UA}(t_1,0) \cdot [\![Q_2]\!]_{UA}(t_2,0) \\ &= (d_1 - c_1) \cdot (d_2 - c_2) \end{aligned}$$

$$\begin{aligned} Q_{join}(t,0,1) &= [\![Q_1]\!]_{UA}(t_1,0) \cdot [\![Q_2]\!]_{UA}(t_2,1) \\ &= (d_1 - c_1) \cdot c_2 \end{aligned}$$

$$\begin{aligned} Q_{join}(t,1,0) &= [\![Q_1]\!]_{UA}(t_1,1) \cdot [\![Q_2]\!]_{UA}(t_2,0) \\ &= c_1 \cdot (d_2 - c_2) \end{aligned}$$

$$\begin{aligned} Q_{join}(t,1,1) &= [\![Q_1]\!]_{UA}(t_1,1) \cdot [\![Q_2]\!]_{UA}(t_2,1) \\ &= c_1 \cdot c_2 \end{aligned}$$

The projection expression $min(Q_1.C, Q_2.C)$ maps the first three cases to $(t,0)$ and the last case to $(t,1)$. Thus,

$$\begin{aligned} Q_{res}(t,0) &= (d_1 - c_1) \cdot (d_2 - c_2) + (d_1 - c_1) \cdot c_2 + c_1 \cdot (d_2 - c_2) \\ &= d_1 \cdot d_2 - c_1 \cdot c_2 \end{aligned}$$

$$Q_{res}(t,1) = c_1 \cdot c_2$$

Finally, we get

$$R'(t) = [d_1 \cdot d_2 - c_1 \cdot c_2 + c_1 \cdot c_2, c_1 \cdot c_2] = [Q_1 \bowtie_\theta Q_2](t)$$

$$
\begin{aligned}
\llbracket R \rrbracket_{UA} &= R \\
\llbracket \sigma_\theta(Q) \rrbracket_{UA} &= \sigma_\theta(\llbracket Q \rrbracket_{UA}) \\
\llbracket \pi_A(Q) \rrbracket_{UA} &= \pi_{A,C}(\llbracket Q \rrbracket_{UA}) \\
\llbracket Q_1 \bowtie_\theta Q_2 \rrbracket_{UA} &= \pi_{\textsc{Sch}(Q_1 \bowtie Q_2), min(Q_1.C, Q_2.C) \to C}(\llbracket Q_1 \rrbracket_{UA} \bowtie_\theta \llbracket Q_2 \rrbracket_{UA}) \\
\llbracket Q_1 \cup Q_2 \rrbracket_{UA} &= \llbracket Q_1 \rrbracket_{UA} \cup \llbracket Q_2 \rrbracket_{UA}
\end{aligned}
$$

Fig. 7: Relational algebra rewrite rules implementing $\llbracket Q \rrbracket_{UA}$

Projection $\pi_A(Q_1)$: $\llbracket \pi_A(Q_1) \rrbracket_{UA} = \pi_{A,C}(\llbracket Q_1 \rrbracket_{UA})$. Recall the definition of projection: $[\pi_A(R_1)](t) = \sum_{s[A]=t} R_1(s)$. Consider a tuple t and let $\{s_1, \ldots, s_n\}$ be the set of tuples with $s_i[A] = t$. Furthermore, let $s_i = [d_i, c_i]$ and

$$R_1{}' = \textsc{Enc}^{-1}(\pi_{A,C}(\textsc{Enc}(R_1))$$

Then, $R_1{}'(t) = [k_{t,0} + k_{t,1}, k_{t,1}]$ for

$$
\begin{aligned}
k_{t,0} &= \sum_{(s,0)[A,U]=(t,0)} \textsc{Enc}(R_1)(s,0) = \sum_{i=1}^{n} d_i - \sum_{i=1}^{n} c_i \\
k_{t,1} &= \sum_{(s,1)[A,U]=(t,1)} \textsc{Enc}(R_1)(s,1) = \sum_{i=1}^{n} c_i
\end{aligned}
$$

Thus, we get

$$
\begin{aligned}
R_1{}'(t) &= [k_{t,0} + k_{t,1}, k_{t,1}] \\
&= \left[\sum_{i=1}^{n} d_i - \sum_{i=1}^{n} c_i + \sum_{i=1}^{n} c_i, \sum_{i=1}^{n} c_i\right] \\
&= \sum_{s[A]=t} R_1(s) \\
&= \pi_A(R_1)(t)
\end{aligned}
$$

Union $Q_1 \cup Q_2$: $\llbracket Q_1 \cup Q_2 \rrbracket_{UA} = (\llbracket Q_1 \rrbracket_{UA} \cup \llbracket Q_2 \rrbracket_{UA})$. Consider a tuple t with $R_i(t) = [d_i, c_i]$. Let

$$R' = \textsc{Enc}^{-1}(\textsc{Enc}(R_1) \cup \textsc{Enc}(R_2))$$

.

We have $[R_1 \cup R_2](t) = [d_1 + d_2, c_1 + c_2]$ and

$$
\begin{aligned}
&[\textsc{Enc}(R_1) \cup \textsc{Enc}(R_2)](t,0)) = (d_1 - c_1) + (d_2 - c_2) \\
&[\textsc{Enc}(R_1) \cup \textsc{Enc}(R_2)](t,1)) = c_1 + c_2
\end{aligned}
$$

Based on this we get

$$
\begin{aligned}
R_1{}'(t) &= [(d_1 - c_1) + (d_2 - c_2) + (c_1 + c_2), c_1 + c_2] \\
&= [d_1 + d_2, c_1 + c_2] \\
&= [d_1, c_1] \oplus_{\mathbb{N}_{UA}} [d_2, c_2] = [R_1 \cup R_2](t)
\end{aligned}
$$

□

B. SQL Implementations of Labeling Schemes

We now show SQL implementations of our methods for extracting best guess worlds and labeling schemes from Section V.

TI-DBs. Consider a TI-DB relation $R(A_1, \ldots, A_n)$ which is stored as a relation $R'(A_1, \ldots, A_n, P)$ where attribute P stores the probabilities of tuples. Recall that we include all tuples t where $P(t) \geq 0.5$ in the best guess world and our labeling scheme for TI-DBs annotates tuples t with T (certain) if $P(t) = 1$. In SQL this is expressed as

```
SELECT A1, ... An,
       CASE WHEN P = 1
            THEN 1
            ELSE 0
       END AS C
FROM R
WHERE P >= 0.5
```

We expect the user to specify the name of the attribute storing the probability for any relation that is marked to be a TI-DB relation. The example shown below illustrates how to mark a relation R which stores probabilities in attribute p as a TI-DB relation.

```
SELECT * FROM R IS TI WITH PROBABILITY (p)
```

x-DBs. For an x-relation $R(A_1, \ldots, A_n)$ which is stored as a relation $R'(X_{id}, Alt_{id}, A_1, \ldots, A_n, P)$ where X_{id} stores identifiers for x-tuples and Alt_{id} store an identifier for alternatives that is unqiue within the scope of an x-tuple. For each x-tuple τ we pick the alternative with the highest probability if the total probability mass of the x-tuple is larger or equal to 0.5. We only mark alternatives of x-tuples as certain if $P(\tau) = 1$ and $|\tau| = 1$. In the SQL implementation we make extensive use of analytical functions (SQL's `OVER`-clause).

```
SELECT A1, ..., An
       CASE WHEN P = 1
            THEN 1
            ELSE 0
       END AS C
FROM R
WHERE Aid = (FIRST_VALUE(Aid)
                OVER (PARTITION BY Xid
                      ORDER BY P DESC))
      AND (1 - sum(P)) OVER (PARTITION BY Xid)
          >=
          max(P) OVER (PARTITION BY Xid)
```

When an input relation is identified as an x-relation, we require that the user specifies which attributes stores x-tuple identifies, alternative identifiers, and probabilities. For example, consider the SQL snipplet shown below.

```
SELECT *
FROM R IS X WITH XID (tid)
                 ALTID (aid)
                 PROBABILITY (p)
```

C-tables. For a C-table $R(A_1, \ldots, A_n)$ which is stored as a relation $R'(A_1, \ldots, A_n, V_1, \ldots, V_n, LC)$ where LC stores the local condition $\phi_{\mathcal{D}}(t)$ (as a string) and V_i stores a variable name if $A_i = v$ for some variable v and **NULL** otherwise. The SQL implementation of the labeling schema and best guess world computation for C-tables assumes the existence of a UDF `isTautology` that implements the tautology check as described in Section V.

```
SELECT A1, ..., An
       CASE WHEN isTautology(LC)
            THEN 1
            ELSE 0
       END AS C
FROM R
WHERE V1 IS NULL AND ... AND Vn IS NULL
```

To mark an input as a C-table the user has to specify which attributes store the V_i's and local condition.

```
SELECT *
FROM R IS CTABLE WITH VARIABLES (V1, ..., Vn)
                      LOCAL CONDITION (lc)
```

IX. Related Work

Incomplete and probabilistic data models. Uncertainty has been recognized as an important problem by the database community early-on. Codd [9] extended the relational model with null values to represent missing information and proposed to use 3-valued logic to evaluate queries over databases with null values. Imielinski [23] introduced V-tables and C-tables as representations of incompleteness. C-tables are closed under full relational algebra. Reiter [34] proposed to model databases as logical theories, a model equivalent to V-tables. Abiteboul [1] defined update operations over incomplete databases. Underlying all these models is the possible world semantics. Probabilistic data models quantify the uncertainty in incomplete databases by assigning probabilities to individual possible worlds. TI-DBs [36] are a prevalent model for probabilistic data where each tuple is associated with its marginal probability and tuples are assumed to be independent. Green et al. [20] studied probabilistic versions of C-tables. Virtual C-tables generalize C-tables [25], [39] by allowing symbolic expressions as values.

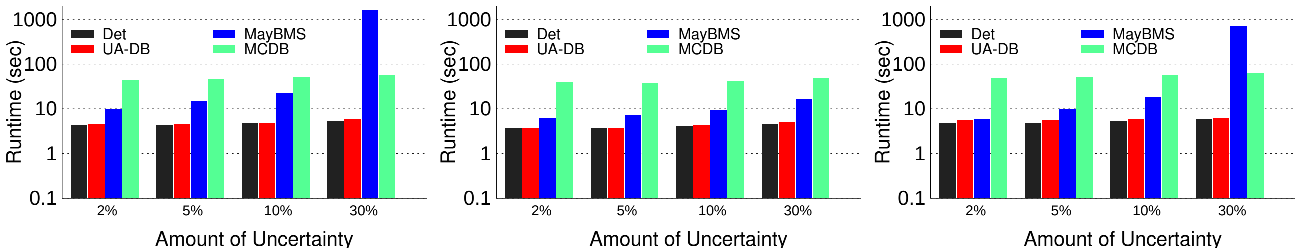
Probabilistic Query Processing. Probabilistic query processing (PQP) has been a field of research for several decades (e.g., an important survey is [36]). Computing the marginal probability of a query result tuple can be reduced to weighted model counting and, thus, is `#P` in general [11]. Most practical approaches for PQP are either limited to queries which can be answered in `PTIME` (so-called *safe* queries) and/or compute approximate probabilities for query answers (e.g., [33]). Systems implementing PQP include Sprout [13], Trio [3], MCDB [24], Mimir [32], MYSTIQ [6], and many others. With the exception of Mimir [32], the above systems demand significant changes on the backend database engine or entirely new database kernels to deploy. Similar to Mimir, our approach is a middleware on top of existing database systems and our implementation does not require any modification to the backend system.

Certain Answers. Many approaches for answering queries over incomplete databases employ certain answer semantics [23], [2], [30], [21], [22]. The foundational work by Lipski [31] defined certain answers analogously to our approach, but using minima instead of GLBs. Computing certain answers is coNP-complete [2], [23] (data complexity) for first order queries, even for restricted data models such as Codd-tables. Thus, it is not surprising that approaches for approximating the set of certain answers have been proposed. Reiter [34] proposed a `PTIME` algorithm that returns a subset of the certain answers (c-sound) for positive existential queries (and a limited form of universal queries). Guagliardo and Libkin [22], [30], [21] propose a query semantics that preserves c-soundness for full relational algebra (first order queries) over Codd- and V-tables. [22] defined the certain and possible multiplicity for tuples under bag semantics and presents some initial thoughts on how to extend the approach from [21] for bag semantics. Sundarmurthy et al. [37] introduced m-tables which can represent not just uncertainty, but also model information about missing tuples. This approach works for both set and bag semantics. The terms c-soundness and c-correctness which we utilize here were coined in this work. Geerts et al. [16] study first order under-approximations of certain answers in the context of data cleaning (PK repairs).

Annotated Databases. Green et al. [18] introduced the semiring annotation framework that we utilize in this work. The connection between annotated databases, provenance, and uncertainty has been recognized early-on. A particular type of semiring annotations, called Lineage in the probabilistic database literature, have been used extensively for probabilistic query processing (e.g., see [36], [35]). Green et al. [18] observed that set semantics incomplete databases can be expressed as $\mathcal{K}$-relations by annotating each tuple with the set of worlds that contain this tuple. We define a more general type of incomplete databases based on $\mathcal{K}$-relations which is defined for any l-semiring. Kostylev et al. [26] investigate how to deal with dependencies among annotations from multiple domains. Similar to [26], we consider "multi-dimensional" annotations, however, for a very different purpose: to extend incomplete databases beyond set semantics.

X. Experiments

We evaluate the performance of queries over UA-DBs on a commercial DBMS for which the licensing restrictions prevent



(a) PDBench - Q1 (b) PDBench - Q2 (c) PDBench - Q3

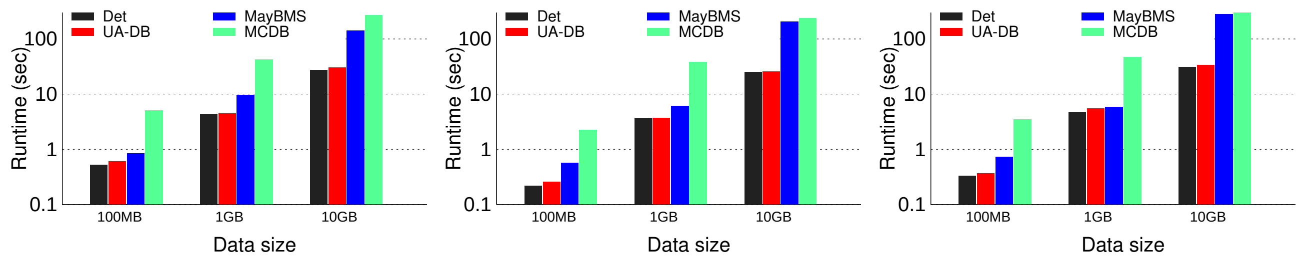
Fig. 8: Performance of PDBench queries - varying the amount of uncertainty

	UA-DB			MayBMS		
	$Q1$	$Q2$	$Q3$	$Q1$	$Q2$	$Q3$
2%	14,260	152,583	9,016	113,966	210,996	15,108
5%	34,041	152,432	8,619	501,114	327,052	32,438
10%	61,800	152,389	8,794	2,392,916	618,199	97,454
30%	130,581	152,885	7,994	134,054,635	3,941,554	4,351,782

Fig. 9: Query result sizes (#rows)

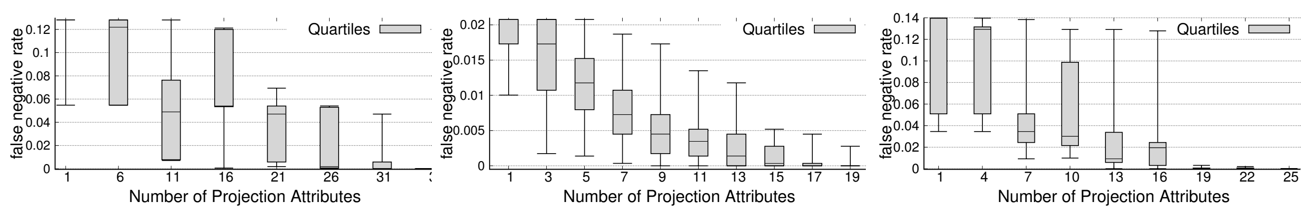
	$Q1$	$Q2$	$Q3$
2%	0 (0%)	143,618 (94%)	7,861 (87%)
5%	1 (0%)	130,594 (86%)	6,023 (70%)
10%	4 (0%)	111,120 (73%)	3,979 (45%)
30%	1 (0%)	52,724 (34%)	586 (7%)

Fig. 10: Result certain answer %

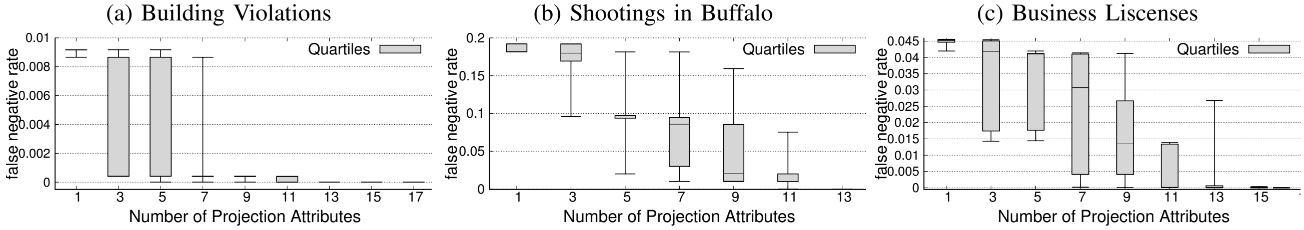


(a) PDBench - Q1 (b) PDBench - Q2 (c) PDBench - Q3

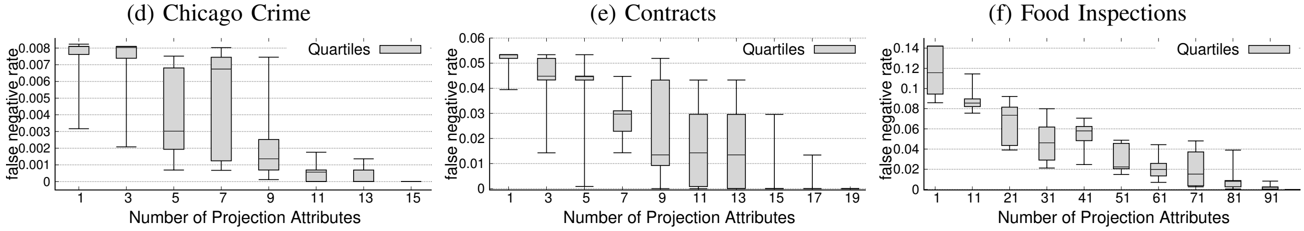
Fig. 11: Performance of PDBench queries - varying database size



(a) Building Violations (b) Shootings in Buffalo (c) Business Liscenses



(d) Chicago Crime (e) Contracts (f) Food Inspections



(g) Graffiti Removal (h) Building Permits (i) Public Library Survy

Fig. 12: Measuring incompleteness as the fraction of certain answers that were misclassified as uncertain

us from identifying the system here. We compare against deterministic best guess query processing and state-of-the-art PQP methods (*MayBMS* and *MCDB*). All experiments are run on a machine with 2×6 core AMD Opteron 4238 CPUs, 128GB RAM, 4×1TB 7.2K HDs (RAID 5). We report the average running time of 5 runs. Furthermore, we also evaluate the false negative rate, i.e., the fraction of certain answers that are misclassified as uncertain by our approach.

A. Performance Comparison Using Synthetic Data

We use PDBench [4] which uses a modified TPC-H data generator [10] that introduces uncertainty by generating random possible values for randomly selected cells (attribute values). The generator produces a columnar encoding of the TPC-H tables as pairs of tuple identifiers and attribute values. An uncertain cell is represented by having multiple attribute values for the same tuple identifier in a column. We directly run MayBMS queries on these columnar tables. For MCDB, we simulate the tuple bundle query using 10 samples. We run deterministic queries and queries generated by our approach on one possible world that is selected by randomly choosing a value for each uncertain cell. For our approach, we treat the input as an x-DB and, thus, mark any tuple with at least one uncertain cell as uncertain. The three PDBench queries used here do roughly correspond to TPC-H queries Q3, Q6 and Q7.

Amount of uncertainty. To evaluate scalability as a function of the amount of uncertainty in the data, we use a scale factor 1 database (∼1GB of data per possible world) and generate versions of this dataset with an uncertainty percentage of 2%, 5%, 10% and 30% (percentage of cells that are uncertain). Each uncertain cell has up to 8 possible values. Figure 8 shows the runtime results for PDBench queries Q1, Q2 and Q3. We observe that, as expected, the runtime for UA-DBs is close to deterministic query processing. The slight overhead of UA-DBs is due to propagating uncertainty annotations. MCDB runs more than 10 times slower than deterministic query processing with sample size of 10 on our testing queries. Again this is expected, since we essentially have to evaluate each query multiple times. MayBMS has reasonable but still noticeable overhead for lower amounts of uncertainty. For larger uncertainty percentages, the overhead over UA-DBs can be several orders of magnitude (the plots use log scale), especially for Q1 and Q3 which involve expensive joins. MayBMS performs better for the simple selection query Q2.

To better understand the performance of MayBMS, we show result sizes (number of tuples) for each query and varying amounts of uncertainty in Figure 9. Our approach produces the same number of results as deterministic processing if applied to the same possible world. MayBMS returns the full set of possible answers and, thus the result size increases dramatically (combinatorial explosion) with increasing amount of uncertainty leading to the system's poor scalability in the amount of uncertainty. We also show the percentage of certain answers for each query per input uncertainty level in Figure 10. The unexpected increase of result size for Q1 over UA-DBs is caused by a shift in the correlation between attributes `o_orderkey` and `l_shipdate` which in turn affects the number of tuples that fulfill Q1's selection condition. This shift is the result of PDBench choosing values for uncertain cells independently.

Dataset size. To evaluate scalability in terms of data size, we use datasets with scale factors (SF) 0.1 (100MB), 1 (1GB) and 10 (10GB) and fix the uncertainty percentage (2%). Figure 11 shows the results for PDBench queries Q1, Q2 and Q3. Again UA-DBs exhibit performance similar to deterministic queries and MCDB is again roughly 10 times slower (the sample size is 10). MayBMS's relative overhead over deterministic processing increases with data set size. For instance, for Q1 the overhead is ∼ 60% for SF 0.1 and ∼500% for SF 10.

B. Real world datasets

We use multiple real world datasets representing a wide variety of domains to evaluate how our approach performs for real world data. We use Mimir [39] to impute missing values in the datasets. In Mimir, imputing missing values introduces attribute-level uncertainty - the cells (attribute values) which are imputed are uncertain. We represent the output as x-relations where $P(\tau) = 1$ for every x-tuple τ. We use labeling scheme $\mathsf{label}_{\text{x-DB}}$ (Section V) which marks all tuples as uncertain that contain at least one uncertain attribute value. Figure 13 shows some basic statistics for the cleaned datasets and URLs for the original datasets: the number of rows, number of attributes, the percentage of attribute values that are uncertain (U_{attr}), and the percentage of rows marked as uncertain by our c-complete labeling scheme (U_{row}).

Incompleteness. To measure the false negative rate of our approach, we use queries that project on a randomly chosen set of attributes. The rationale for this is that based on Theorem 5 when we project an uncertain tuple onto a subset of its certain attribute attributes (no x-key), then this tuple is a certain answers (no matter what values are chosen for its uncertain attributes, the tuple will be in the result). However, our approach would misclassify it as uncertain since it is (correctly) marked as uncertain in the input. We evaluate queries which project on a randomly chosen set of attributes and measure the false negative rate. Figure 12a to 12i show the distribution of the false negative rate (min, 25-percentile, median, 75-percentile, max) for queries with a fixed number of projection attributes. As expected, the false negative rate decreases with increasing number of projection attributes, but is quite low in general (less than 20% in the worst case for the worst case dataset). For most of the datasets, the median false negative rate is below 5% when at least half of the attributes are involved in the projection. Note that selection and join would not produce any "new" false negative results (see proof of Theorem 5). This shows that for real world datasets with correlated errors the false negative rate is typically low.

C. Real Queries

Furthermore, we evaluate the effectiveness of our approach for five real queries over the real world datasets (the SQL code and descriptions of these queries are shown below). Most of

Dataset	Rows	Cols	U_{Attr}	U_{Row}	URL
Building Violations	1.3M	35	0.82%	12.8%	https://data.cityofchicago.org/Buildings/Building-Violations/22u3-xenr
Shootings in Buffalo	2.9K	21	0.24%	2.1%	http://projects.buffalonews.com/charts/shootings/index.html
Business Licenses	63K	25	1.39%	14.0%	https://data.cityofchicago.org/Community-Economic-Development/Business-Licenses-Current-Active/uupf-x98q
Chicago Crime	6.6M	17	0.21%	0.9%	https://data.cityofchicago.org/Public-Safety/Crimes-2001-to-present/ijzp-q8t2
Contracts	94K	13	1.50%	19.2%	https://data.cityofchicago.org/Administration-Finance/Contracts/rsxa-ify5
Food Inspections	169K	16	0.34%	4.6%	https://data.cityofchicago.org/Health-Human-Services/Food-Inspections/4ijn-s7e5
Graffiti Removal	985K	15	0.09%	0.8%	https://data.cityofchicago.org/Service-Requests/311-Service-Requests-Graffiti-Removal/hec5-y4x5
Building Permits	198K	19	0.42%	5.3%	https://https://www.kaggle.com/aparnashastry/building-permit-applications-data/data
Public Library Survy	9.2K	99	1.19%	14.2%	https://www.imls.gov/research-evaluation/data-collection/public-libraries-survey/explore-pls-data/pls-data

Fig. 13: Real World Datasets

	$Q1$	$Q2$	$Q3$	$Q4$	$Q5$
Overhead	2.28%	1.81%	1.32%	2.88%	3.51%
Error Rate	0.55%	0.37%	0%	0.92%	0.29%

Fig. 14: Real Query Results

our real world datasets are from the City of Chicago Data Portal (https://data.cityofchicago.org/) which also associates analyses (e.g., visualizations) with datasets. The queries we use here were reverse engineered from these analyses. We measure both the performance overhead of UA-DBs and the false negative rate. Performance overheads are measured as relative increase in run time over deterministic query processing. As Figure 14 shows, our approach introduces slight (less than 4%) run time overhead for these queries. Q5 involves a join operator which has an overhead close to 4% while the overhead for the other queries containing only selections and projections was less than 3%. All query results have less than 1% false negative rate. Q3 has relatively small result size and, thus, returns no misclassified results.

Q1. This query is expressed over the Chicago crime dataset. The query returns all crime ids and case numbers for all thefts, domestic batteries, and criminal damages. Here, attribute `IUCR` (Illinois Uniform Crime Reporting code) is a system for specifying crime types.

```
SELECT id, case_number,
       CASE IUCR
         WHEN 0820 then 'Theft'
         WHEN 0486 then 'Domestic␣Battery'
         WHEN 1320 then 'Criminal␣Damage'
       END AS crime_type
FROM Q
WHERE IUCR=0820 OR IUCR=0486 OR IUCR=1320
```

Q2. Find all crime ids, case numbers, longitudes and latitudes of crimes within a retangular area containing Chicago Water-Tower.

```
SELECT id, case_number, Longitude, Latitude
FROM crime
WHERE Longitude BETWEEN -87.674 AND -87.619
      AND Latitude BETWEEN 41.892 AND 41.903
```

Q3. This is a query over the graffiti dataset. Q3 returns all street addresses and zip codes for graffiti removal requests that are currently open.

```
SELECT Street_Address, ZIP_Code, status
FROM graffiti
WHERE status='Open'
```

Q4. Find all dates, addresses and zip codes of food inspections of restaurants that passed, but were identified as “high risk”.

```
SELECT Inspection_Date, address, zip
FROM foodinspections
WHERE results = 'Pass␣w/␣Conditions'
      AND risk = 'Risk␣1␣(High)'
```

Q5. For each crime id, case numbers, and IUCRs of crimes, find all status, service request numbers and community areas from graffiti removal requests where both take place in district 8 and the graffiti removal request's location is within 100 coordinate units of the crime's location.

```
SELECT c.ID,
       c.Case_Number,
       c.IUCR,
       g.status,
       g.Service_Request_Number,
       g.Community_Area
FROM
     (SELECT * FROM graffiti
      WHERE police_district = 8) g,
     (SELECT * FROM crime
      WHERE district = '008') c
WHERE c.X_Coordinate < g.X_Coordinate + 100
  AND c.X_Coordinate > g.X_Coordinate - 100
  AND c.Y_Coordinate < g.Y Coordinate + 100
  AND c.Y_Coordinate > g.Y_Coordinate - 100
```

XI. Conclusions and Future Work

We propose UA-DBs as a novel way to represent uncertainty as annotations on the tuples of a possible world. Being based on $\mathcal{K}$-relations, our approach applies to the incomplete version of any data model that can be encoded as $\mathcal{K}$-relations including set and bag semantics. We achieve efficient query evaluation over UA-DBs by sacrificing completeness (we may mark certain tuples as uncertain), but not soundness (we never mark uncertain tuples as certain). In future work, we plan to extend our approach with attribute level annotations to encode certainty at finer granularity and to support larger classes of queries, e.g., queries involving negation and aggregation.

References

[1] Serge Abiteboul and Gösta Grahne. Update semantics for incomplete databases. In *VLDB*, pages 1–12, 1985.

[2] Serge Abiteboul, Paris C. Kanellakis, and Gösta Grahne. On the representation and querying of sets of possible worlds. *Theor. Comput. Sci.*, 78(1):158–187, 1991.

[3] Parag Agrawal, Omar Benjelloun, Anish Das Sarma, Chris Hayworth, Shubha U. Nabar, Tomoe Sugihara, and Jennifer Widom. Trio: A system for data, uncertainty, and lineage. In *VLDB*, pages 1151–1154, 2006.

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

[5] L. Antova, C. Koch, and D. Olteanu. MayBMS: Managing incomplete information with probabilistic world-set decompositions. In *ICDE*, pages 1479–1480, 2007.

[6] Jihad Boulos, Nilesh N. Dalvi, Bhushan Mandhani, Shobhit Mathur, Christopher Ré, and Dan Suciu. MYSTIQ: a system for finding more answers by using probabilities. In *SIGMOD*, pages 891–893, 2005.

[7] Stanley Burris and H.P. Sankappanavar. *A Course in Universal Algebra*. Springer, 2012.

[8] Xu Chu, John Morcos, Ihab F. Ilyas, Mourad Ouzzani, Paolo Papotti, Nan Tang, and Yin Ye. Katara: A data cleaning system powered by knowledge bases and crowdsourcing. In *SIGMOD*, pages 1247–1261, 2015.

[9] E. F. Codd. Extending the database relational model to capture more meaning. *TODS*, 4(4):397–434, 1979.

[10] Transaction Processing Performance Council. TPC-H specification. http://www.tpc.org/tpch/.

[11] Nilesh Dalvi and Dan Suciu. The dichotomy of conjunctive queries on probabilistic structures. In *PODS*, pages 293–302, 2007.

[12] Ronald Fagin, Benny Kimelfeld, and Phokion G. Kolaitis. Probabilistic data exchange. *J. ACM*, 58(4):15:1–15:55, 2011.

[13] Robert Fink, Andrew Hogue, Dan Olteanu, and Swaroop Rath. Sprout^2: a squared query engine for uncertain web data. In *SIGMOD*, pages 1299–1302, 2011.

[14] Robert Fink, Jiewen Huang, and Dan Olteanu. Anytime approximation in probabilistic databases. *VLDBJ*, 22(6):823–848, 2013.

[15] Wolfgang Gatterbauer and Dan Suciu. Dissociation and propagation for approximate lifted inference with standard relational database management systems. *VLDBJ*, 26(1):5–30, 2017.

[16] Floris Geerts, Fabian Pijcke, and Jef Wijsen. First-order under-approximations of consistent query answers. *International Journal of Approximate Reasoning*, 83:337–355, 2017.

[17] Floris Geerts and Antonella Poggi. On database query languages for k-relations. *J. Applied Logic*, 8(2):173–185, 2010.

[18] Todd J. Green, Grigoris Karvounarakis, and Val Tannen. Provenance semirings. In *PODS*, pages 31–40, 2007.

[19] Todd J. Green, Grigoris Karvounarakis, Nicholas E. Taylor, Olivier Biton, Zachary G. Ives, and Val Tannen. ORCHESTRA: Facilitating collaborative data sharing. In *SIGMOD*, pages 1131–1133, 2007.

[20] Todd J. Green and Val Tannen. Models for incomplete and probabilistic information. *IEEE Data Eng. Bull.*, 29(1):17–24, 2006.

[21] Paolo Guagliardo and Leonid Libkin. Making SQL queries correct on incomplete databases: A feasibility study. In *PODS*, pages 211–223, 2016.

[22] Paolo Guagliardo and Leonid Libkin. Correctness of SQL queries on databases with nulls. *SIGMOD Record*, 46(3):5–16, 2017.

[23] Tomasz Imielinski and Witold Lipski Jr. Incomplete information in relational databases. *J. ACM*, 31(4):761–791, 1984.

[24] Ravi Jampani, Fei Xu, Mingxi Wu, Luis Leopoldo Perez, Christopher Jermaine, and Peter J Haas. MCDB: a monte carlo approach to managing uncertain data. In *SIGMOD*, pages 687–700, 2008.

[25] O. Kennedy and C. Koch. Pip: A database system for great and small expectations. In *ICDE*, pages 157–168, 2010.

[26] Egor V. Kostylev and Peter Buneman. Combining dependent annotations for relational algebra. In *ICDT*, pages 196–207, 2012.

[27] Willis Lang, Rimma V. Nehme, Eric Robinson, and Jeffrey F. Naughton. Partial results in database systems. In *SIGMOD*, pages 1275–1286, 2014.

[28] Julie Letchner, Christopher Ré, Magdalena Balazinska, and Matthai Philipose. Access methods for markovian streams. In *ICDE*, pages 246–257, 2009.

[29] Jian Li, Barna Saha, and Amol Deshpande. A unified approach to ranking in probabilistic databases. *VLDBJ.*, 20(2):249–275, 2011.

[30] Leonid Libkin. SQL's three-valued logic and certain answers. *TODS*, 41(1):1:1–1:28, 2016.

[31] Witold Lipski. On semantic issues connected with incomplete information databases. *TODS*, 4(3):262–296, 1979.

[32] Arindam Nandi, Ying Yang, Oliver Kennedy, Boris Glavic, Ronny Fehling, Zhen Hua Liu, and Dieter Gawlick. Mimir: Bringing ctables into practice. *CoRR*, abs/1601.00073, 2016.

[33] Dan Olteanu, Jiewen Huang, and Christoph Koch. Approximate confidence computation in probabilistic databases. In *ICDE*, pages 145–156, 2010.

[34] Raymond Reiter. A sound and sometimes complete query evaluation algorithm for relational databases with null values. *J. ACM*, 33(2):349–370, 1986.

[35] Anish Das Sarma, Martin Theobald, and Jennifer Widom. Exploiting lineage for confidence computation in uncertain and probabilistic databases. In *ICDE*, pages 1023–1032, 2008.

[36] Dan Suciu, Dan Olteanu, Christopher Ré, and Christoph Koch. Probabilistic databases. *Synthesis Lectures on Data Management*, 3(2):1–180, 2011.

[37] Bruhathi Sundarmurthy, Paraschos Koutris, Willis Lang, Jeffrey F. Naughton, and Val Tannen. m-tables: Representing missing data. In *ICDT*, pages 1–20, 2017.

[38] Moshe Y. Vardi. Querying logical databases. *J. Comput. Syst. Sci.*, 33(2):142–160, 1986.

[39] Ying Yang, Niccolò Meneghetti, Ronny Fehling, Zhen Hua Liu, and Oliver Kennedy. Lenses: An on-demand approach to etl. *PVLDB*, 8(12):1578–1589, 2015.