Approximating Certain and Possible Answers beyond Sets
Logic and Algebra for Query Evaluation, Logic and Algorithms in Database Theory and AI

Boris Glavic

DBGroup
Illinois Institute of Technology

2023-11-14
Motivation

Incomplete & Probabilistic Databases

- Principled techniques for managing uncertain data
- High computational complexity for query answering
  - Many tractable cases based on query / data structure are known [DS12, DS07, FO16, FHO12, RS09, ABS15]
  - intractable even for relatively simple queries
**Definition (Incomplete Database)**

An incomplete database is a set $\mathcal{D} = \{D_1, \ldots, D_n\}$ where each $D_i$, called a *possible world*, is a deterministic database.

**Example**

<table>
<thead>
<tr>
<th>name</th>
<th>country</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boris</td>
<td>USA</td>
</tr>
<tr>
<td>Peter</td>
<td>USA</td>
</tr>
<tr>
<td>Alexandra</td>
<td>USA</td>
</tr>
</tbody>
</table>

$D_1$

<table>
<thead>
<tr>
<th>name</th>
<th>country</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boris</td>
<td>Switzerl.</td>
</tr>
<tr>
<td>Alexandra</td>
<td>USA</td>
</tr>
</tbody>
</table>

$D_2$

<table>
<thead>
<tr>
<th>name</th>
<th>country</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boris</td>
<td>Germany</td>
</tr>
<tr>
<td>Peter</td>
<td>Netherl.</td>
</tr>
<tr>
<td>Alexandra</td>
<td>USA</td>
</tr>
</tbody>
</table>

$D_3$
Definition (Possible World Semantics)

- Incomplete Databases

\[ Q(\mathcal{D}) = \{ Q(D) \mid D \in \mathcal{D} \} \]
Definition (Possible Tuples)

Possible(\(D\)) = \{ t \mid \exists D \in D : t \in D \} 

Definition (Certain Tuples)

Certain(\(D\)) = \{ t \mid \forall D \in D : t \in D \} 

Definition (Possible & Certain Answers)

Certain(\(Q, D\)) = Certain(Q(\(D\)))

Possible(\(Q, D\)) = Possible(Q(\(D\)))
A practical data model for uncertain data management

1. Supports bags (in addition to sets)
2. PTIME data complexity (through approximation)
3. Support for complex queries (full relational algebra with aggregation, sort-based operations, recursion)
4. Modular: closed under queries
5. Compact: be able to trade size for approximation accuracy
6. Compatible: approximate existing uncertain data models and data cleaning paradigms (consistent query answering)
Outline

1. Incomplete Databases, Certain & Possible Answers
2. Incomplete K-relations
3. Approximating Incomplete K-relations
4. Attribute-level Uncertainty
5. Conclusions & Future Work
6. Appendix and References
Motivation

- Incomplete Databases beyond sets
  - Incomplete bag databases
  - Incomplete provenance
  - Incomplete temporal databases
  - Incomplete access control
  - ...
**Semirings**

- **semiring** \((K, +_K, \cdot_K, 0_K, 1_K)\)
- a set \(K\)
- binary operations \(+_K\) and \(\cdot_K\) which are associative and commutative
- \(0_K (1_K)\) are the neutral elements of \(+_K\) and \(\cdot_K\)
- \(k \cdot_K 0_K = 0_K\) for all \(k \in K\)
- \(k_1 \cdot_K (k_2 +_K k_3) = (k_1 \cdot_K k_2) + (k_1 \cdot_K k_3)\)

**Semiring-annotated relations**

- An n-ary \(K\)-relation \(R\) is a function \(\mathbb{U}^n \rightarrow K\) with finite support: \(\{t \mid R(t) \neq 0_K\}\) ([GKT07, GT17])
- Tuples that are annotated with \(0_K\) do not exist (omit them)
Natural order

- \( k_1 \leq_K k_2 \) if exists \( k_3 \) such that \( k_1 + k_3 = k_2 \)
- Semirings where the natural order has a lattice structure are called \( l \)-semirings [KB12]
  - Each pair of elements \( k_1 \) and \( k_2 \) has ...
    - a greatest lower bound \( \sqcap_K(k_1, k_2) \)
    - a least upper bound \( \sqcup_K(k_1, k_2) \)

Natural order of \( \mathbb{N} \)

- \( k_1 \leq_{\mathbb{N}} k_2 \) is the standard order of natural numbers (e.g., \( 2 < 3 \))
- \( \sqcap_{\mathbb{N}}(k_1, k_2) = \min(k_1, k_2) \)
- \( \sqcup_{\mathbb{N}}(k_1, k_2) = \max(k_1, k_2) \)
Definition (Incomplete $K$-database)

An incomplete $K$-database is a set $\mathcal{D} = \{D_1, \ldots, D_n\}$ such that each $D_i$ is a $K$-database.

Possible world semantics

- Possible world semantics uses $K$-relational query semantics

Certain and possible answers

- Consider only $I$-semirings
- The certain annotation of a tuple
  - $\text{Cert}_K(\mathcal{D}, t) = \bigcap_K \{D_i(t) \mid D_i \in \mathcal{D}\}$
- The possible annotation of a tuple
  - $\text{Poss}_K(\mathcal{D}, t) = \bigcup_K \{D_i(t) \mid D_i \in \mathcal{D}\}$

Coincides with certain / possible tuples for sets / bags
Incomplete Bag Relations

Bag Semantics Example (\(\mathbb{N}\) annotations)

- Annotations encode the multiplicity of tuples under bag semantics
- Possible annotation is \(\text{max}\), certain annotation is \(\text{min}\)
- The certain (possible) annotation of \((Peter, USA)\) is \(2 (3)\)
- The certain (possible) annotation of \((Boris, USA)\) is \(0 (2)\)

\[D_1\]

<table>
<thead>
<tr>
<th>name</th>
<th>country</th>
<th>(\mathbb{N})</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boris</td>
<td>USA</td>
<td>2</td>
</tr>
<tr>
<td>Peter</td>
<td>USA</td>
<td>3</td>
</tr>
</tbody>
</table>

\[D_2\]

<table>
<thead>
<tr>
<th>name</th>
<th>country</th>
<th>(\mathbb{N})</th>
</tr>
</thead>
<tbody>
<tr>
<td>Peter</td>
<td>USA</td>
<td>2</td>
</tr>
</tbody>
</table>
1. Incomplete Databases, Certain & Possible Answers
2. Incomplete K-relations
3. Approximating Incomplete K-relations
4. Attribute-level Uncertainty
5. Conclusions & Future Work
6. Appendix and References
Definition (Containment)
For two $K$-database $D_1$ and $D_2$ over the same schema, $D_1$ is contained in $D_2$ ($D_1 \subseteq_K D_2$) iff
\[ \forall t : D_1(t) \leq_K D_2(t) \]

Definition (Under-approximating certain annotations)
A $K$-database $D$ is an under-approximation of an incomplete $K$-database $D$ iff:
\[ D \subseteq_K \text{Certain}(D) \]

Definition (Over-approximating possible annotations)
A $K$-database $D$ is an over-approximation of an incomplete $K$-database $D$ iff:
\[ D \supseteq_K \text{Possible}(D) \]
A tuple-independent database $D = D_{certain} \cup D_{possible}$ encodes an incomplete $\mathbb{B}$-database with the possible worlds $Mod(D)$:

$$Mod(D) = \{ D' \mid D_{certain} \subseteq_\mathbb{B} D' \land D' \subseteq_\mathbb{B} D \}$$

**Approximating TIDBs**

- **Under-approximation**: $D = D_{certain}$
- **Over-approximation**: $D = D$
Consider a \( B \)-relation \( R(A_1,\ldots,A_n) \) with a key \( K \subseteq \{A_1,\ldots,A_n\} \) and the incomplete database of all S-repairs:

\[
R = \{ R' \mid R' \subseteq R \land \forall k \in \pi_K(R) : \exists t \in R' : t.K = k \\
\quad \land \lnot t_1 \neq t_2 \in R' : t_1.K = t_2.K \}
\]

**Approximating key repairs**

- **under-approximation:**

\[
R' = \{ t \mid R(t) = \top \land \lnot t' \neq t : t.K = t'.K \land R(t') = \top \}
\]

- **over-approximation:**

\[
R' = R
\]
Definition (Uncertainty-annotated databases)

- A UA-DB \( D \) is a \( K_{AU} \)-relation where \( K_{AU} \) is the product semiring \( K \times K \) restricted to \( \{(k_l, k_u) \mid k_l \leq_k k_2\} \).
- Addition and multiplication in \( K^2 \) are defined point-wise.
- Define \((c, p)^\downarrow = c\) and \((c, p)^\uparrow = p\) and lift this operation pointwise to databases.
  - \( D^\downarrow(t) = D(t)^\downarrow \) and \( D^\uparrow(t) = D(t)^\uparrow \).

Proposition (Structure \( K_{AU} \) is a semiring)

For any naturally-ordered semiring \( K \), \( K_{AU} \) is a semiring.
Bounding incomplete $K$-databases

An $K$-UA-DB $D$ bounds an incomplete $K$-database $\mathcal{D}$ ($D \preceq_K \mathcal{D}$) iff:

$$D \downarrow \subseteq_K \text{Certain}(\mathcal{D})$$

$$D \uparrow \supseteq_K \text{Possible}(\mathcal{D})$$

Bound-preserving query semantics

A query semantics for $K_{AU}$ for is called bound-preserving iff for any query $Q$:

$$D \preceq_K \mathcal{D} \Rightarrow Q(D) \preceq_K Q(\mathcal{D})$$
Theorem (Positive relational algebra is bound-preserving)

For any positive relational algebra query $Q$, standard $K$-relational query semantics is bound preserving.

Proof Sketch.

Addition and multiplication are monotone wrt. the natural order. If $k_1 \leq_K k_3$ and $k_2 \leq_K k_4$ then:

- $k_1 +_K k_2 \leq_K k_3 +_K k_4$
- $k_1 \cdot_K k_2 \leq_K k_3 \cdot_K k_4$
Theorem (Full relational algebra is bound-preserving)

Full relational algebra is bound-preserving assuming the monus-semiring definition from [GP10] and by defining:

\[(R - S)(t) = (R(t)\downarrow - \kappa S(t)\uparrow, R(t)\uparrow - \kappa S(t)\downarrow)\]
Recursive Datalog queries over UA-DBs are bound preserving.

As computations in $D^\downarrow$ and $D^\uparrow$ are standard K-relational query evaluation, existing results for Datalog over K-relations are applicable. Specifically, we get the same convergence guarantees as proven for K-relations in [KNP+22].
Consistent query answering & certain answers

- The following approach is bound preserving:
  - Answer a supported subquery of a query $Q$ using an existing technique for certain / possible answers
  - Interpret the result as a UA-DB
  - Evaluate the remainder of the query using UA-DBs
Outline

1. Incomplete Databases, Certain & Possible Answers
2. Incomplete K-relations
3. Approximating Incomplete K-relations
4. Attribute-level Uncertainty
5. Conclusions & Future Work
6. Appendix and References
Motivation

- Certain answers are often empty
  - e.g., operations like aggregation that return different aggregation function results in each world
- Some types of uncertainty cannot be efficiently over-approximated with "concrete" tuples
  - This person certainly exists, but their salary is in \([100000, 200000]\)
- Operations like \(\text{sum}\) aggregation may produce exponentially many possible results
Related work

- Representation systems that use (labelled) nulls
  - C-tables [ILJ84], m-tables [SKL+17], Codd-tables, V-tables, ...

- "Factorized" representations
  - OR-tables [IVV95], x-tables (block-independent databases) [BSHW06, VdBS17]

- Returning the lowest and highest possible aggregation function result
  - [ABC+03, Fux07, ACK+10, LSV02, AK08]

- Abstract interpretation
  - [CC77, CC04]
Interval domains

- Domain \((D, \leq_D)\) where \(\leq_D\) is a partial order
- Interval domain \(D^\leq = \{[l, u] | l, u \in D \land l \leq_D u\}\)
- Representation system \(Mod([l, u]) = \{c | c \in [l, u]\}\)

Over-approximating operations

- \([l_1, u_1] + [l_2, u_2] := [l_1 + l_2, u_1 + u_2]\)
- \([l_1, u_1] = [l_2, u_2] := [l_1 = u_1 = l_2 = u_2, l_1 \leq u_2 \land l_2 \leq u_1]\)
The AU-DB Model

AU-DB relations

- Special type of $K$-relations
  - Value domains are intervals
  - Annotation domain is a UA-semiring $K_{AU}$

The possible worlds encoded by a UA-DB

- A UA-DB $D$ encodes as possible worlds $\text{Mod}(D)$ every K-database $D$ such that we can "redistribute" the annotations of tuples from $D$ to bounding tuples from $D$ such that
  - for any tuple $t$ from $D$, the total annotation mass distributed to $t$ is within its annotation bounds
### UA-DB Example

#### UA-DB

<table>
<thead>
<tr>
<th>name</th>
<th>salary</th>
<th>$N^3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boris</td>
<td>[120k,120k]</td>
<td>[0,2]</td>
</tr>
<tr>
<td>Peter</td>
<td>[140k,400k]</td>
<td>[2,3]</td>
</tr>
</tbody>
</table>

#### Two possible worlds

<table>
<thead>
<tr>
<th>name</th>
<th>salary</th>
<th>$N$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boris</td>
<td>120k</td>
<td>1</td>
</tr>
<tr>
<td>Peter</td>
<td>150k</td>
<td>1</td>
</tr>
<tr>
<td>Peter</td>
<td>380k</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>name</th>
<th>salary</th>
<th>$N$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Peter</td>
<td>180k</td>
<td>2</td>
</tr>
</tbody>
</table>
Bounding incomplete databases

- Given an incomplete $\mathbb{N}$ database $\mathcal{D}$ and selected world $D_{sg} \in \mathcal{D}$
- Create UA-DB $\mathcal{D}$ with selected-guess $D_{sg}$ such that $\text{Mod}(\mathcal{D}) \supseteq \text{Mod}(\mathcal{D})$

Incomplete bag database

<table>
<thead>
<tr>
<th></th>
<th>name</th>
<th>salary</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$D_1$</td>
<td>Boris</td>
<td>120k</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Peter</td>
<td>400k</td>
<td>3</td>
</tr>
<tr>
<td>$D_2$</td>
<td>Peter</td>
<td>140k</td>
<td>2</td>
</tr>
</tbody>
</table>

A Bounding UA-DB for SG $D_1$

<table>
<thead>
<tr>
<th></th>
<th>name</th>
<th>salary</th>
<th>$\mathbb{N}^3$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Boris</td>
<td>[120k,120k]</td>
<td>[0,2]</td>
</tr>
<tr>
<td></td>
<td>Peter</td>
<td>[140k,400k]</td>
<td>[2,3]</td>
</tr>
</tbody>
</table>
Alternative Interpretation of Bounding

Bounding
- Given a UA-DB $D$ that bounds an incomplete $K$-database $D$

Under-approximation of certain tuples with attribute uncertainty
- The lower bound tuple annotations of $D$
  - Under-approximation of certain tuples
- Generalizes certain answers by allowing attribute-level uncertainty

Over-approximation of possible tuples with attribute uncertainty
- The upper bound tuple annotations of $D$
  - Over-approximation of possible tuples
- Generalizes possible answers by allowing attribute-level uncertainty
Query Semantics for $\mathcal{RA}^+$

Queries over $K$-relations
- Defined using the semiring addition / multiplication operations
- Disjunctive use of tuples (union, projection) is addition
- Conjunctive use of tuples (join) is multiplication

$\mathcal{RA}^+$ Semantics

Union: $(R_1 \cup R_2)(t) = R_1(t) +_K R_2(t)$

Join: $(R_1 \bowtie R_2)(t) = R_1(t[Sch(R)_1]) \cdot_\text{K} R_2(t[Sch(R)_2])$

Projection: $(\pi_U(R))(t) = \sum_{t=t'[U]} R(t')$

Selection: $(\sigma_\theta(R))(t) = R(t) \cdot_\text{K} \theta(t)$
Conditional expressions over range-annotated values

- Expressions over range-annotated values evaluate to a range of Boolean values
  - e.g., $[1, 3] = [3, 3]$ evaluates to $[F, T]$

Evaluating selections

- in $K$-relations selection on $\theta$
  - $false$ maps to $0_K$ and $true$ to $1_K$
  - tuples get their input annotation if they do fulfill $\theta$
  - tuples get a $0_K$ annotation in the result if they do not fulfill $\theta$
- in $K_{AU}$ relations: apply this to every dimension
  - e.g., $[F, T]$ maps to $[0, 1]$
Semi-module Aggregation?

**Semi-modules**
- Polynomial representation system for aggregation results [ADT11]

**Proposition** ($\text{SUM}_I$, $\text{MAX}_I$, and $\text{MIN}_I$ are monoids)
- using our definitions of addition, min, max over the interval domain

**Proposition** (Semi-modules cannot be bound preserving)
- Use $\mathbb{N}_{AU}$ and $\text{SUM}_I$, set $k = (1, 2)$ and $m = [0, 0]$ and $m_1 = [-1, -1]$ and $m_2 = [1, 1]$
- Observe that $m = m_1 + m_2$
- Based on semi-module laws we have to have $k \otimes m = k \otimes (m_1 + m_2) = k \otimes m_1 + k \otimes m_2$
- However, $k \otimes m_2 = (l_2, u_2)$ with $l_2 \leq 1 < 2 \leq u_2$
Aggregation over AU-DBs

Escape into uncertainty
- Use bound-preserving multiplication of intervals instead of a semimodule
- \( k \otimes m = k \cdot m \)

Approach
- Group-by attribute bounds in the result determined based on assigning tuples to groups based on their selected-guess values
- Aggregation function result bounds by reasoning about
  - Which tuples could / have to belong to a group
  - Taking the lowest / highest possible multiplicities of tuples into account
  - Using lowest / highest possible values of the attribute we are aggregating over
  - Utilize expression semantics for range-annotated values
Sort Order as Data
- there is only one physical sort order
- encode sort position as UA-DB range-values!

Windowed Aggregation
- exploit fixed window sizes

---

- Certain window
- Possible window
- Not in window
- Certainly in
- Possibly in
Outline

1. Incomplete Databases, Certain & Possible Answers
2. Incomplete K-relations
3. Approximating Incomplete K-relations
4. Attribute-level Uncertainty
5. Conclusions & Future Work
6. Appendix and References
Conclusions

UA-DB Data Model

- Lightweight, but powerful uncertain data model
- Under-approximation of certain answers
- Over-approximation of possible answers

Query semantics

- **Bound preservation**: the bounding of certain and possible answers is preserved under queries
- Ranges (attribute-level uncertainty) as first-class citizens: generalizes
  - certain answers with nulls [LJ84]
  - "bounding range semantics" for aggregation over incomplete data
Future Work

Even more queries
- Recursive and iterative computations (e.g., training ML models)

Even better performance
- New specialized physical operators
- Optimizing representations and queries

Better bounds
- Using more expressive set representations instead of boxes

Beyond ranges
- Partial orders / Ontologies instead of ranges
### Implementations

- **GProM** [https://github.com/IITDBGroup/gprom](https://github.com/IITDBGroup/gprom)
- **Vizier** [https://vizierdb.info/](https://vizierdb.info/)

---

Su Feng  Oliver Kenndy  Aaron Huber
Outline

1. Incomplete Databases, Certain & Possible Answers
2. Incomplete K-relations
3. Approximating Incomplete K-relations
4. Attribute-level Uncertainty
5. Conclusions & Future Work
6. Appendix and References
Experimental Results - TPC-H Queries

Figure: TPC-H (SF=1)
Competitors

- **UA-DB**
  - UA-DB: no over-approximation of possible and no attribute-uncertainty
  - AU-DB: the model discussed so far
- **Under-approximation of certain for DBs with nulls (Libkin, [Lib16])**
- **Sampling-based** (*MCDB w/o probabilities, [JXW⁺08]*)
- **All possible answers**
  - *MayBMS w/o probabilities*\(^a\), [AKO07, OHK09])
  - *Trio w/o probabilities*\(^b\), [BSHW06]
  - *Semi-modules w/o probabilities* [FHO12]

\(^a\)Does not support aggregation
\(^b\)Does not support uncertain group-by and querying aggregation results
### Datasets
- *PD-Bench* (TPC-H data with errors), [AJKO08]
- Real-world open datasets with missing values

### Machines
- 2×6 core AMD Opteron 4238 CPUs
- 128GB RAM
- 4×1TB 7.2K HDDs (RAID 5)
- Postgres 11
Figure: PD-Bench Queries (1GB) - Varying input uncertainty
Figure: Simple aggregation queries over TPC-H (1GB) - Varying number of aggregation operations
Figure: Experiments with Real Datasets (Netflix, Crimes, Healthcare)
<table>
<thead>
<tr>
<th>Datasets &amp; Queries</th>
<th>Time (sec)</th>
<th>cert. tup.</th>
<th>attr. bounds</th>
<th>pos.tup. by id</th>
<th>pos.tup. by val</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Netflix</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AU-DB</td>
<td>0.011</td>
<td>100%</td>
<td>1</td>
<td>1</td>
<td>100%</td>
</tr>
<tr>
<td>Trip</td>
<td>0.900</td>
<td>100%</td>
<td>1</td>
<td>1</td>
<td>100%</td>
</tr>
<tr>
<td>SPJ</td>
<td>0.049</td>
<td>N/A</td>
<td>1</td>
<td>1</td>
<td>99.6%</td>
</tr>
<tr>
<td>UA-DB</td>
<td>0.006</td>
<td>100%</td>
<td>N/A</td>
<td>N/A</td>
<td>99.1%</td>
</tr>
<tr>
<td><strong>Crimes</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AU-DB</td>
<td>0.082</td>
<td>100%</td>
<td>1</td>
<td>4</td>
<td>100%</td>
</tr>
<tr>
<td>Trip</td>
<td>1.700</td>
<td>100%</td>
<td>1</td>
<td>1</td>
<td>98.8%</td>
</tr>
<tr>
<td>SPJ</td>
<td>0.118</td>
<td>N/A</td>
<td>1</td>
<td>1</td>
<td>99.9%</td>
</tr>
<tr>
<td>UA-DB</td>
<td>0.009</td>
<td>0%</td>
<td>N/A</td>
<td>N/A</td>
<td>99.3%</td>
</tr>
<tr>
<td><strong>Healthcare</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AU-DB</td>
<td>1.58</td>
<td>100%</td>
<td>1</td>
<td>1</td>
<td>100%</td>
</tr>
<tr>
<td>Trip</td>
<td>59.0</td>
<td>100%</td>
<td>1</td>
<td>1</td>
<td>100%</td>
</tr>
<tr>
<td>SPJ</td>
<td>6.91</td>
<td>N/A</td>
<td>0.6</td>
<td>1</td>
<td>99.9%</td>
</tr>
<tr>
<td>UA-DB</td>
<td>0.63</td>
<td>100%</td>
<td>N/A</td>
<td>N/A</td>
<td>99.9%</td>
</tr>
<tr>
<td><strong>GB</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AU-DB</td>
<td>2.09</td>
<td>100%</td>
<td>1</td>
<td>1.01</td>
<td>100%</td>
</tr>
<tr>
<td>Trip</td>
<td>103.1</td>
<td>100%</td>
<td>1</td>
<td>1</td>
<td>100%</td>
</tr>
<tr>
<td>SPJ</td>
<td>5.24</td>
<td>N/A</td>
<td>0.99</td>
<td>0</td>
<td>100%</td>
</tr>
<tr>
<td>UA-DB</td>
<td>0.47</td>
<td>0%</td>
<td>N/A</td>
<td>N/A</td>
<td>100%</td>
</tr>
</tbody>
</table>
Lenses

- Uncertainty-aware Data Cleaning Operations [YMF+15]

UA-DBs

- The basic UA-DB model [FHGK19]
- Attribute-level uncertainty, range values, and aggregation over UA-DBs (the AU-DB model) [FHGK21]
- Sorting, top-k, and windowed aggregation [FGK23]

Vizer

- http://www.vizierdb.info
- Publications:
  http://www.cs.iit.edu/~dbgroup/publications.html (click on project Vizier to see relevant publications)
**Relational Encoding**

**Encode UA-DBs as relational databases**
- attribute-level ranges
  - for each attribute \( A \) add \( A_{lb} \) and \( A_{ub} \)
- annotation ranges
  - add attributes \( row_{lb}, row_{sg}, row_{ub} \)

**UA-DB**

<table>
<thead>
<tr>
<th>name</th>
<th>salary</th>
<th>( \mathbb{N}^3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boris</td>
<td>[120k,120k,120k]</td>
<td>[0,2,2]</td>
</tr>
<tr>
<td>Peter</td>
<td>[140k,400k,400k]</td>
<td>[2,3,3]</td>
</tr>
</tbody>
</table>

**Relational encoding**

<table>
<thead>
<tr>
<th>name</th>
<th>salary(_{lb})</th>
<th>salary</th>
<th>salary(_{ub})</th>
<th>row(_{lb})</th>
<th>row(_{sg})</th>
<th>row(_{ub})</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boris</td>
<td>120k</td>
<td>120k</td>
<td>120k</td>
<td>0</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Peter</td>
<td>140k</td>
<td>400k</td>
<td>400k</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>
Implementing UA-DB Query Semantics

Rewrite-based approach

Specialized Operator Implementations
- one pass algorithms for sorting and windowed aggregation

Existing implementations
- GProM (https://github.com/IITDBGroup/gprom)
- Vizier (https://vizierdb.info/)
Optimizations for Aggregation & Join

Performance of Joins and Aggregation

- Aggregation and joins require interval overlap joins
- $O(n^2)$ in many DBMS

Optimizations

- Split possible from selected-guess and process them separately
  - Selected-guess does not require interval overlap joins
  - Compress possible, interval overlap joins over smaller inputs
    - Trade performance for accuracy of the over-approximation
- This works, because approximate compression is bound preserving
Sorting and Windowed Aggregation

**Rewrite-based approach**
- can be expensive: range-joins and multiple sorting steps

**Native Algorithms**
- one pass algorithms
- implemented inside Postgres
- requires reasoning about certain and possible windows and maintain rows in multiple sort orders

**Connected Heaps**

![Diagram of connected heaps with arrows indicating relationships between elements]

**Result of h1.pop()**

![Diagram showing the result of h1.pop() with updated connections between elements]
COVID infection rate data

- COVID infection rate per location from two sources

Query - maximum infection rate per location size

SELECT size, max(rate) as mrate
FROM R GROUP BY size

Source 1

<table>
<thead>
<tr>
<th>city</th>
<th>size</th>
<th>rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Los Angeles</td>
<td>Metro</td>
<td>2%</td>
</tr>
<tr>
<td>Washington</td>
<td>Metro</td>
<td>6%</td>
</tr>
<tr>
<td>Chicago</td>
<td>City</td>
<td>4%</td>
</tr>
<tr>
<td>Springfield</td>
<td>Town</td>
<td>2%</td>
</tr>
</tbody>
</table>

Source 2

<table>
<thead>
<tr>
<th>city</th>
<th>size</th>
<th>rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Los Angeles</td>
<td>Metro</td>
<td>7%</td>
</tr>
<tr>
<td>Washington</td>
<td>Village</td>
<td>8%</td>
</tr>
<tr>
<td>Chicago</td>
<td>City</td>
<td>3%</td>
</tr>
</tbody>
</table>
Running Example - Conflicting Information

Conflicting Information

- Washington may belong to group *Metro* or *Village*
- Conflicting rates for Los Angeles
- Does Springfield exist?

<table>
<thead>
<tr>
<th>Source 1</th>
<th>Source 2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>city</strong></td>
<td><strong>city</strong></td>
</tr>
<tr>
<td>Los Angeles</td>
<td>Los Angeles</td>
</tr>
<tr>
<td>Washington</td>
<td>Washington</td>
</tr>
<tr>
<td>Chicago</td>
<td>Chicago</td>
</tr>
<tr>
<td>Springfield</td>
<td>Springfield</td>
</tr>
<tr>
<td><strong>size</strong></td>
<td><strong>size</strong></td>
</tr>
<tr>
<td>Metro</td>
<td>Metro</td>
</tr>
<tr>
<td>Metro</td>
<td>Village</td>
</tr>
<tr>
<td>City</td>
<td>City</td>
</tr>
<tr>
<td>Town</td>
<td>Town</td>
</tr>
<tr>
<td><strong>rate</strong></td>
<td><strong>rate</strong></td>
</tr>
<tr>
<td>2%</td>
<td>7%</td>
</tr>
<tr>
<td>6%</td>
<td>8%</td>
</tr>
<tr>
<td>4%</td>
<td>4%</td>
</tr>
</tbody>
</table>
Ignoring uncertainty in data leads to hard to trace errors with severe real world consequences.

**Query - maximum infection rate per location size**

```sql
SELECT size, max(rate) as mrate
FROM R GROUP BY size
```

### "Cleaned" Sources

<table>
<thead>
<tr>
<th>city</th>
<th>size</th>
<th>rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Los Angeles</td>
<td>Metro</td>
<td>2%</td>
</tr>
<tr>
<td>Washington</td>
<td>Village</td>
<td>8%</td>
</tr>
<tr>
<td>Chicago</td>
<td>City</td>
<td>4%</td>
</tr>
</tbody>
</table>

### Query Result

<table>
<thead>
<tr>
<th>city</th>
<th>rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Village</td>
<td>8%</td>
</tr>
<tr>
<td>Metro</td>
<td>2%</td>
</tr>
<tr>
<td>City</td>
<td>4%</td>
</tr>
</tbody>
</table>
### Why Not Clean Your Data Upfront?

- **Heuristic algorithms**: pick one repair heuristically
- **Highest probability repair**: pick the most likely repair

### Cleaned Data = Ground Truth?

- The ground truth is typically hard / impossible to determine

### Take-away

Effectively, most data cleaning & integration techniques select one possible repair

- All information about uncertainty is lost!
- How can we trust any analysis result based on the cleaned data?
(R1) Efficiency

- Efficiently generate compact representations of uncertainty
- Running queries over uncertain data should be efficient
- Otherwise, users will just ignore uncertainty
(R1) Efficiency
- Efficiently generate compact representations of uncertainty
- Running queries over uncertain data should be efficient
- Otherwise, users will just ignore uncertainty

(R2) Expressiveness
- Model complex types of uncertainty
- Complex SQL queries / computations
- Unlikely to be adopted otherwise
Requirements for Managing Uncertainty

(R1) Efficiency
- Efficiently generate compact representations of uncertainty
- Running queries over uncertain data should be efficient
- Otherwise, users will just ignore uncertainty

(R2) Expressiveness
- Model complex types of uncertainty
- Complex SQL queries / computations
- Unlikely to be adopted otherwise

(R3) Guarantees
- Provide as much information about uncertainty as possible
- Queries should take uncertainty into account
(R1) Efficiency
- Efficiently generate compact representations of uncertainty
- Running queries over uncertain data should be efficient
- Otherwise, users will just ignore uncertainty

(R2) Expressiveness
- Model complex types of uncertainty
- Complex SQL queries / computations
- Unlikely to be adopted otherwise

(R3) Guarantees
- Provide as much information about uncertainty as possible
- Queries should take uncertainty into account

(R4) Usability
- Interpretable by mere mortals
- Backwards-compatible with existing cleaning & integration solutions
Sort Order as Data
- there is only one physical sort order
- encode sort position as UA-DB range-values!

Windowed Aggregation
- exploit fixed window sizes

Diagram:
- t
  - t
    - t
      - t
        - t
          - t
            - t
              - t
                - t
                  - t
                    - t
                      - t
                        - t
                          - t
                            - t
                              - t
                                - t
                                 - t
                                  - t
                                    - t
                                      - t
                                        - t
                                          - t
                                            - t
                                              - t
                                                - t
                                                  - t
                                                    - t
                                                      - t
                                                        - t
                                                          - t
                                                            - t
                                                              - t
                                                                - t
                                                                  - t
                                                                    - t
                                                                      - t
                                                                        - t

0 1 2 3 4 5 6 7 8 9 10 11 12
<table>
<thead>
<tr>
<th>Queries</th>
<th>2%/SF0.1</th>
<th>2%/SF1</th>
<th>5%/SF1</th>
<th>10%/SF1</th>
<th>30%/SF1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AU-DB</td>
<td>1.607</td>
<td>15.636</td>
<td>15.746</td>
<td>15.811</td>
<td>16.021</td>
</tr>
<tr>
<td>Det</td>
<td>0.560</td>
<td>1.833</td>
<td>1.884</td>
<td>1.882</td>
<td>1.883</td>
</tr>
<tr>
<td>Q3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AU-DB</td>
<td>0.713</td>
<td>7.830</td>
<td>8.170</td>
<td>8.530</td>
<td>7.972</td>
</tr>
<tr>
<td>Det</td>
<td>0.394</td>
<td>1.017</td>
<td>1.058</td>
<td>1.092</td>
<td>1.175</td>
</tr>
<tr>
<td>MCDB</td>
<td>4.112</td>
<td>11.138</td>
<td>11.222</td>
<td>10.936</td>
<td>11.454</td>
</tr>
<tr>
<td>Q5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AU-DB</td>
<td>0.846</td>
<td>8.877</td>
<td>8.803</td>
<td>8.839</td>
<td>8.925</td>
</tr>
<tr>
<td>Det</td>
<td>0.247</td>
<td>0.999</td>
<td>1.012</td>
<td>1.123</td>
<td>1.117</td>
</tr>
<tr>
<td>MCDB</td>
<td>2.599</td>
<td>10.152</td>
<td>10.981</td>
<td>11.527</td>
<td>11.909</td>
</tr>
<tr>
<td>Q7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AU-DB</td>
<td>0.791</td>
<td>7.484</td>
<td>7.537</td>
<td>7.303</td>
<td>7.259</td>
</tr>
<tr>
<td>Det</td>
<td>0.145</td>
<td>0.977</td>
<td>0.985</td>
<td>0.989</td>
<td>1.044</td>
</tr>
<tr>
<td>MCDB</td>
<td>1.472</td>
<td>10.123</td>
<td>10.277</td>
<td>10.749</td>
<td>10.900</td>
</tr>
<tr>
<td>Q10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AU-DB</td>
<td>0.745</td>
<td>7.377</td>
<td>7.283</td>
<td>7.715</td>
<td>8.012</td>
</tr>
<tr>
<td>Det</td>
<td>0.263</td>
<td>1.024</td>
<td>0.993</td>
<td>1.004</td>
<td>1.015</td>
</tr>
<tr>
<td>MCDB</td>
<td>2.691</td>
<td>10.743</td>
<td>10.937</td>
<td>11.826</td>
<td>11.697</td>
</tr>
</tbody>
</table>
**Experimental Results - Ranking + Windows**

- **MCDB** [JXW+08] - sample-based approach (10 or 20 samples)
- **PT-k** [HPZL08] - Full certain / possible answers for top-k queries
- **Symb** - Computing certain (possible) answers with constraint solver

### Datasets & Queries

<table>
<thead>
<tr>
<th>Datasets &amp; Queries</th>
<th>Imp (time)</th>
<th>Det (time)</th>
<th>MCDB20 (time)</th>
<th>Rewr (time)</th>
<th>Symb (time)</th>
<th>PT-k (time)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Iceberg</strong> [ice07]</td>
<td>0.816ms</td>
<td>0.123ms</td>
<td>2.337ms</td>
<td>1.269ms</td>
<td>278ms</td>
<td>1s</td>
</tr>
<tr>
<td>(1.1%, 167K)</td>
<td>Window</td>
<td>2.964ms</td>
<td>0.363ms</td>
<td>7.582ms</td>
<td>1.046ms</td>
<td>589ms</td>
</tr>
<tr>
<td><strong>Crimes</strong> [crigo]</td>
<td>1043.505ms</td>
<td>94.306ms</td>
<td>2001.12ms</td>
<td>14787.723ms</td>
<td>&gt;10min</td>
<td>&gt;10min</td>
</tr>
<tr>
<td>(0.1%, 1.45M)</td>
<td>Window</td>
<td>3.050ms</td>
<td>0.416ms</td>
<td>8.337ms</td>
<td>2.226ms</td>
<td>&gt;10min</td>
</tr>
<tr>
<td><strong>Healthcare</strong> [heare]</td>
<td>287.515ms</td>
<td>72.289ms</td>
<td>1451.232ms</td>
<td>4226.260ms</td>
<td>15s</td>
<td>8s</td>
</tr>
<tr>
<td>(1.0%, 171K)</td>
<td>Window</td>
<td>130.496ms</td>
<td>15.212ms</td>
<td>323.911ms</td>
<td>13713.218ms</td>
<td>&gt;10min</td>
</tr>
</tbody>
</table>

### Datasets & Measures

<table>
<thead>
<tr>
<th>Datasets &amp; Measures</th>
<th>Imp/Rewr</th>
<th>MCDB20</th>
<th>PT-k/Symb</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Iceberg</strong> [ice07]</td>
<td>bound accuracy</td>
<td>0.891</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>bound recall</td>
<td>1</td>
<td>0.765</td>
</tr>
<tr>
<td><strong>Crimes</strong> [crigo]</td>
<td>bound accuracy</td>
<td>0.996</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>bound recall</td>
<td>1</td>
<td>0.919</td>
</tr>
<tr>
<td><strong>Healthcare</strong> [heare]</td>
<td>bound accuracy</td>
<td>0.990</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>bound recall</td>
<td>1</td>
<td>0.767</td>
</tr>
</tbody>
</table>
• **Open-source**: https://vizierdb.info/

• **A Data-centric Workflow Engine Instead of a REPL**
  - no hidden state!
  - incremental workflow execution

• **A Hybrid Notebook + Spreadsheet UI**
  - build workflows incrementally through a notebook interface
  - edit datasets as spreadsheets

• **Caveats: Data Concerns that Propagate**
  - document data errors / concerns / uncertainty
  - caveats propagate alongside the data
  - supports quantification of data uncertainty using UA-DBs
  - summaries as overview of problems with a workflow

• **Workflow Provenance and Versioning**
  - Versioning (workflow evolution provenance)
Data-centric Workflows

Datasets
- Currently relational tables
- Spreadsheet interpretation (co-ordinate system)

Workflow Steps (cells)
- Cell executions are isolated from each other
- Dataflow through Vizier’s dataset API
  - Cells consume and produce datasets
Vizier Architecture

Figure: Vizier’s architecture
The Notebook UI

Projects consists of a notebook & datasets

A notebook is a sequence of cells
- Cells are steps in a workflow
- User-facing interface like Jupyter or similar
- Cells run in isolation and communicate through datasets

Datasets
- Datasets are created / read / updated by cells
- Dataset creation
  - Loading (special cell type)
  - Result of a cell (e.g., SQL cell)
- Dataset update
  - Spreadsheet operations (e.g., update value)
  - Curation cells (e.g., impute missing values)
The Spreadsheet UI & Vizual

Spreadsheet UI
- Updates are translated into workflow steps

Spreadsheets as Datasets
- Relation + coordinate system (cell locations)

Vizual language
- Scripting language for common spreadsheet operations

Figure: Column data distributions shown in the spreadsheet view
Workflow Evolution and Versioning

Workflow versioning

- Vizier captures provenance for the evolution of a user’s workflow
- Like Git, but automates versioning
  - Every edit creates a new version
  - Branching is supported
- Versions of the workflow and of datasets have unique URLs
## Incremental Execution model

### Immutable objects
- Cells & their outputs are immutable objects (at least conceptually)

### Dependency tracking
- Cell ordering is sequential (position in the notebook)
- Dependencies are tracked automatically
  - Monitoring Vizier dataset API calls

### Automatic refresh
- On cell update, we determine automatically which cells are dependent and need to be re-executed
Caveats: Uncertainty and Error Handling

Caveats

- Caveats are annotations on data (attribute value)
- Record concerns / document assumptions about data
- May list possible alternative values or may mark value as unknown (uncertainty)
  - UA-DBs without boolean attribute markers instead of ranges

Caveat Generation

- Users can manually create caveats
- Data curation operations can annotated their outputs with caveats
Repairing a violated PK constraint

- **Primary key**: Name
- **Key repair**: \( \text{avg(Age)} \)
- Repaired value is caveated (highlighted in red)
  - Equivalent to annotating with a range that covers the whole domain

<table>
<thead>
<tr>
<th>Name</th>
<th>Age</th>
</tr>
</thead>
<tbody>
<tr>
<td>Peter</td>
<td>30</td>
</tr>
<tr>
<td>Peter</td>
<td>35</td>
</tr>
<tr>
<td>Bob</td>
<td>25</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Age</th>
</tr>
</thead>
<tbody>
<tr>
<td>Peter</td>
<td>32.5</td>
</tr>
<tr>
<td>Bob</td>
<td>25</td>
</tr>
</tbody>
</table>
## UI & API Support for Caveats

### Dataset error summary
- Per dataset
- Shows counts of caveats grouped by type

### Spreadsheet view
- Caveated values are shown in red
- Users can inspect caveat

### Plots
- Bars / points based on caveated values are shown with a red dot

### Dataset API / SQL / Lenses
- Functions for accessing / creating / manipulating caveats
- SQL functions to create / check for caveats
Cell types - curation & cleaning

Data curation (lenses)
- Missing value imputation
- Pivot
- Geocoding
- Type detection
- Schema Matching
- ...

Uncertainty-aware through UA-DBs
- Results stored as UA-DBs


References II


[GKT07] Todd J. Green, Gregory Karvounarakis, and Val Tannen. 
Provenance Semirings. 

On database query languages for K-relations. 

The semiring framework for database provenance. 

[heare] Medicare hospital dataset. 

[HPZL08] Ming Hua, Jian Pei, Wenjie Zhang, and Xuemin Lin. 
Ranking queries on uncertain data: a probabilistic threshold approach. 

[ice07] Iceberg dataset. 
https://nsidc.org/data/g00807.

[ILJ84] Tomasz Imieliński and Witold Lipski Jr. 
Incomplete Information in Relational Databases. 


Sprout: Lazy vs. eager query plans for tuple-independent probabilistic databases.

[RS09] Christopher Ré and Dan Suciu.
The trichotomy of HAVING queries on a probabilistic database.

[SKL+17] Bruhathi Sundarmurthy, Paraschos Koutris, Willis Lang, Jeffrey Naughton, and Val Tannen.
m-tables: Representing missing data.

Query processing on probabilistic data: A survey.
2017.

[YMF+15] Ying Yang, Niccolo Meneghetti, Ronny Fehling, Zhen Hua Liu, and Oliver Kennedy.
Lenses: an on-demand approach to etl.