OLAP Over Uncertain and Imprecise Data

Doug Burdick[‡] Prasad M. Deshpande[*] T.S. Jayram[*]
Raghu Ramakrishnan[‡] Shivakumar Vaithyanathan [*]

[*] IBM Almaden Research Center [‡] University of Wisconsin, Madison

Abstract

We extend the OLAP data model to represent data ambiguity, specifically imprecision and uncertainty, and introduce an allocation-based approach to the semantics of aggregation queries over such data. We identify three natural query properties and use them to shed light on alternative query semantics. While there is much work on representing and querying ambiguous data, to our knowledge this is the first paper to handle both imprecision and uncertainty in an OLAP setting.

1 Introduction

In this paper, we extend the multidimensional OLAP data model to represent *data ambiguity*, specifically *imprecision* and *uncertainty*, and study possible semantics for aggregation queries over such data. While there is much work on representing and querying ambiguous data, and even some work in the context of OLAP, to our knowledge this is the first paper to identify criteria that must be satisfied by any approach to handling data ambiguity in an OLAP setting, and to use these criteria in a principled manner to arrive at appropriate semantics for queries. Our first criterion, called *consistency*, accounts for the relationship between similar queries issued at related nodes in a domain hierarchy in order to meet users' intuitive expectations as they navigate up and down the hierarchy. The second criterion, called *faithfulness*, captures the intuition that more precise data should lead to better results. The third criterion, called *correlation-preservation*, essentially requires that the statistical properties of the data should not be affected by the allocation of ambiguous data records. While the last two criteria are not specific to OLAP, to our knowledge they have not been proposed previously.

We extend the usual OLAP data model in two fundamental ways. First, we relax the restriction that dimension attributes in a fact must be assigned leaf-level values from the underlying domain hierarchy, in order to model *imprecision*. For example, we can denote that a particular repair took place in Texas, without specifying a city. Clearly, this has implications for how we answer queries—for a query that aggregates repair costs in Austin, should the example repair be included, and if so, how? Our second extension is to introduce a new kind of measure attribute that represents uncertainty. Intuitively, an uncertain value encodes a range of possible values together with our belief in the likelihood of each possible value. Specifically, we represent a value for an uncertain measure as a *probability distribution function (pdf)* over values from an associated "base" domain.

Our contributions can be summarized as follows:

1. Generalization of the OLAP model to represent data ambiguity. To our knowledge, this is the first such generalization that addresses both imprecise dimension values and uncertain measure values.
2. The introduction of criteria (consistency, faithfulness, correlation-preservation) that guide the choice of semantics for aggregation queries over ambiguous data.
3. A possible-worlds interpretation of data ambiguity that leads to a novel allocation-based approach to defining semantics for aggregation queries, and a careful study of choices arising in the treatment of data ambiguity, using the consistency, faithfulness, and correlation-preservation criteria.
4. Algorithms for evaluating aggregation queries (including AVERAGE, COUNT, and SUM for ordinary measures, and LinOp for uncertain measures), together with a complexity analysis.
5. An experimental evaluation that addresses scalability as well as result quality.

1.1 Related Work

While there is an extensive literature on queries over ambiguous data, only a few papers [24, 28, 29, 16] have considered an OLAP setting. [24] is perhaps the closest to our work in that it considers the semantics of aggregation queries, but it does not consider uncertainty or investigate criteria that shed light on appropriate query semantics.

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the VLDB copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Very Large Data Base Endowment. To copy otherwise, or to republish, requires a fee and/or special permission from the Endowment.

Proceedings of the 31st VLDB Conference,
Trondheim, Norway, 2005

[28, 29] consider the problem of imputing missing measure values, using linear equations, entropy maximization, cross-entropy minimization, or other constraint programming techniques. [16] describes a method to estimate the error of cube aggregates for certain data; the generalization of their method to uncertain data is not clear.

The earliest work on aggregate queries over imprecise data is [8], and it was followed by [20, 9, 25]; however, none of these papers consider data-level hierarchies (which are central to OLAP). The approach in [8] leads to an exponential-time algorithm for SUM. [9] models uncertainty using intervals and pdfs and provides algorithms for aggregating them. [25] develops a linear programming based semantics for computing aggregates over probabilistic databases. We note that [20] also discusses uncertainty, and [26] supports aggregate functions for uncertain data, but doesn't support imprecision or hierarchies.

We believe that a key contribution of this paper is our methodology—identifying intuitive criteria such as consistency, faithfulness, and correlation-preservation and using them to study alternative query semantics is an approach that can be applied outside the OLAP setting (and indeed, faithfulness and correlation-preservation are not specific to OLAP). In this respect, our approach is similar in spirit to the use of *summarizability*, which was introduced to study the interplay between *properties of data* and the aggregation operators (i.e., what properties should the data possess for results of certain aggregation functions to be meaningful) [18, 19].

A number of papers consider imprecision and uncertainty for non-aggregation queries. The use of possible world semantics to represent imprecise data is discussed in [1]. [7, 13, 4, 17, 12] associate a probability distribution with the data (either at the data element or tuple level), and generalize relational algebra operators to reflect the associated probability. [2, 3] seek to identify inconsistent data and to "repair" these databases to a consistent state; in contrast, we focus on imprecise yet consistent data, and do not consider integrity constraints (other than domain constraints). Various sources of data ambiguity are classified in [22, 23], together with approaches for representing and processing the ambiguity. [27] discusses the many similarities between statistical databases and OLAP.

2 Data Model

In this section we present our generalization of the standard multidimensional data model, incorporating imprecision and uncertainty.

2.1 Data Representation

Attributes in the standard OLAP model are of two kinds—*dimensions* and *measures*. We extend the model to support uncertainty in measure values and imprecision in dimension values.

Definition 1 (Uncertain Domains)**.** An *uncertain domain* U over base domain O is the set of all possible probability distribution functions, or pdfs, over O. □

Thus, each value u in U is a pdf that, intuitively, indicates our degree of belief that the "true" value being represented is o, for each o in the base domain O. For an example of uncertain domains we consider in this paper, see Section 2.2.1.

Definition 2 (Imprecise Domains)**.** An *imprecise domain* I over a base domain B is a subset of the powerset of B with $\emptyset \notin I$; elements of I are called *imprecise* values. □

Intuitively, an imprecise value is a non-empty set of possible values. Allowing dimension attributes to have imprecise domains enables us, for example, to use the imprecise value `Wisconsin` for the location attribute in a data record if we know that the sale occurred in Wisconsin but are unsure about the city.

In OLAP, each dimension has an associated hierarchy, e.g., the location dimension might have attributes *City* and *State*, with *State* denoting generalizations of *City*; this suggests a natural special case of imprecise domains called *hierarchical* domains, which we define next.

Definition 3 (Hierarchical Domains)**.** A *hierarchical* domain H over base domain B is defined to be an imprecise domain over B such that (1) H contains every singleton set (i.e., corresponds to some element of B) and (2) for any pair of elements $h_1, h_2 \in H$, $h_1 \supseteq h_2$ or $h_1 \cap h_2 = \emptyset$. □

Intuitively, each singleton set is a leaf node in the domain hierarchy and each non-singleton set in H is a non-leaf node; thus, `Madison`, `Milwaukee,` etc. are leaf nodes with parent `Wisconsin` (which, in turn might have USA as its parent). We will often refer to a hierarchical domain in terms of leaf and non-leaf nodes, for convenience.

Definition 4 (Fact Table Schemas and Instances)**.** A *fact table schema* is $\langle A_1, A_2, \ldots, A_k; M_1, \ldots, M_n\rangle$ where (i) each dimension attribute $A_i, i \in 1 \ldots k$, has an associated domain dom(A_i) that is *imprecise*, and (ii) each measure attribute $M_j, j \in 1 \ldots n$, has an associated domain dom(M_j) that is either *numeric* or *uncertain*.

A *database instance* of this fact table schema is a collection of *facts* of the form $\langle a_1, a_2, \ldots, a_k; m_1, \ldots, m_n\rangle$ where $a_i \in$ dom$(A_i), i \in 1 \ldots k$ and $m_j \in$ dom$(M_j), j \in 1 \ldots n$. In particular, if dom(A_i) is hierarchical, a_i can be any leaf or non-leaf node in dom(A_i). □

Definition 5 (Regions and Cells)**.** Consider a fact table schema with dimension attributes $A_1, A_2, \ldots, A_k$. A vector $\langle c_1, c_2, \ldots, c_k\rangle$ is called a *cell* if every c_i is an element of the base domain of A_i, $i \in 1 \ldots k$. The *region* of a *dimension vector* $\langle a_1, a_2, \ldots, a_k\rangle$ is defined to be the set of cells $\{\langle c_1, c_2, \ldots, c_k\rangle \mid c_i \in a_i, i \in 1 \ldots k\}$. Let reg$(r)$ denote the region associated with a fact r. □

Proposition 1. *Consider a fact table schema with dimension attributes $A_1, A_2, \ldots, A_k$ that all have hierarchical domains. Consider a k-dimensional space in which each*

	Auto	*Loc*	*Repair*	*Text*	*Brake*
p1	F-150	NY	\$200	...	$\langle 0.8, 0.2\rangle$
p2	F-150	MA	\$250	...	$\langle 0.9, 0.1\rangle$
p3	F-150	CA	\$150	...	$\langle 0.7, 0.3\rangle$
p4	Sierra	TX	\$300	...	$\langle 0.3, 0.7\rangle$
p5	Camry	TX	\$325	...	$\langle 0.7, 0.3\rangle$
p6	Camry	TX	\$175	...	$\langle 0.5, 0.5\rangle$
p7	Civic	TX	\$225	...	$\langle 0.3, 0.7\rangle$
p8	Civic	TX	\$120	...	$\langle 0.2, 0.8\rangle$
p9	F150	East	\$140	...	$\langle 0.5, 0.5\rangle$
p10	Truck	TX	\$500	...	$\langle 0.9, 0.1\rangle$

Table 1: Sample data in a CRM application for automobiles

axis i is labeled with the leaf nodes of dom(A_i). For every region, the set of all cells in the region is a contiguous k-dimensional hyper-rectangle that is orthogonal to the axes.

If every dimension attribute has a hierarchical domain, we thus have an intuitive interpretation of each fact in the database as a region in a k-dimensional space. If all a_i are leaf nodes, the observation is *precise*, and describes a region consisting of a single cell. If one or more A_i are assigned non-leaf nodes, the observation is *imprecise* and describes a larger k-dimensional region. Each cell inside this region represents a possible completion of an imprecise fact, formed by replacing non-leaf node a_i with a leaf node from the subtree rooted at a_i. The process of completing every imprecise fact in this manner represents a *possible world* for the database (Section 4).

2.2 Motivating Example

Consider the scenario of a car manufacturer using a CRM application to track and manage service requests across its worldwide dealer operations. A fact table illustrating such data is shown in Table 1. Each fact describes an "incident". The first two columns are dimension attributes *Automobile* (*Auto*) and *Location* (*Loc*), and take values from their associated hierarchical domains. The structure of these domains and the regions of the facts are shown in Figure 1. Precise facts, p1–p8 in Table 1, have leaf nodes assigned to both dimension attributes and are mapped to the appropriate cells in Figure 1. Facts p9 and p10, on the other hand, are imprecise. Fact p9 is imprecise because the *Location* dimension is assigned to the non-leaf node `East` and its region contains the cells `(NY,F150)` and `(MA,F150)`. Similarly, the region for p10 contains the cells `(TX,F150)` and `(TX,Sierra)`. Each fact contains a value for the numeric measure attribute *Repair* denoting the repair cost associated with the incident.

2.2.1 Example of Measure Uncertainty

We want to classify incidents based on the type of problem (e.g., "brake", "transmission", "engine noise" etc.), as described in the auxiliary *Text* attribute. The subjective nature of text precludes the unambiguous classification of service reports into the different problem types. We can model this ambiguity by defining an uncertain measure whose values are represented as pdfs over the set of problem types. However, due to the dynamic nature of text analysis engines, new problem types will be continuously added. Therefore, it is impractical to assume that the base domain of problem types can be fixed apriori.

To address this, we assume there exist trained classifiers for each type of problem (e.g., see [30]) that output a discrete probability distribution based on analyzing the content of the *Text* attribute; the pdf output reflects the uncertainty inherent in such classifiers. In the example, the output of the classifier for the brake topic is represented as a pdf over two values `Yes` and `No`, and is stored in the uncertain measure attribute *Brake* as a pair of probabilities. (as shown in Table 1). *However, we note that all analysis and algorithms presented henceforth are applicable to attributes with base domain greater than 2.*

2.3 Queries

While the OLAP paradigm offers a rich array of query operators, the basic query consists of selecting a node for one or more dimensions and applying an aggregation operator to a particular measure attribute. For example, selecting the Location node `TX` and the Automobile node `Civic` and applying SUM to the *Repair* measure returns the total amount spent on repairs of `Civic` cars in Texas. All other queries (such as *roll-up, slice, drill-down, pivot*, etc.) can be described in terms of repeated applications of basic queries. We therefore concentrate on studying the semantics of basic queries in light of our two data model extensions; the extension to the full array of OLAP query operators is straightforward, and is omitted for lack of space.

Definition 6 (Queries and Query Results)**.** A query Q over a database D with schema $\langle A_1, A_2, \ldots, A_k; M_1, \ldots, M_n\rangle$ has the form $Q(a_1, \ldots, a_k; M_i, \mathcal{A})$, where: (i) $a_1, \ldots, a_k$ describes the k-dimensional region being queried, (ii) M_i describes the measure of interest, and (iii) $\mathcal{A}$ is an aggregation function.

The result of Q is obtained by applying $\mathcal{A}$ to a set of facts FIND-RELEVANT$(a_1, \ldots, a_k, D)$. This is described in detail below. □

The function FIND-RELEVANT identifies the set of facts in D deemed "relevant" to the query region, and the appropriate definition of this function is an important issue addressed in this paper. All precise facts within the query region are naturally included, but we have an important design decision with respect to imprecise facts. We have three options: ignore all imprecise facts (the None option), include only those contained in the query region (the Contains option), or include all imprecise facts whose region overlaps (i.e., intersects) the query region (Overlaps option).

2.4 Motivating Example (Contd.)

How to handle imprecise facts when answering queries is a central issue, which we now illustrate through an example. In later sections, we study the various options for determining the facts relevant to a query more rigorously. We consider aggregate queries of the type *"What are the repair costs for F150's in the East"?*, i.e., a SUM aggregate value for the measure attribute *Repair* in the region denoted by `(F150,East)`. All queries are depicted in Figure 1 as boxes enclosing the query region. For instance, the above example query corresponds to Q5 in Figure 1.

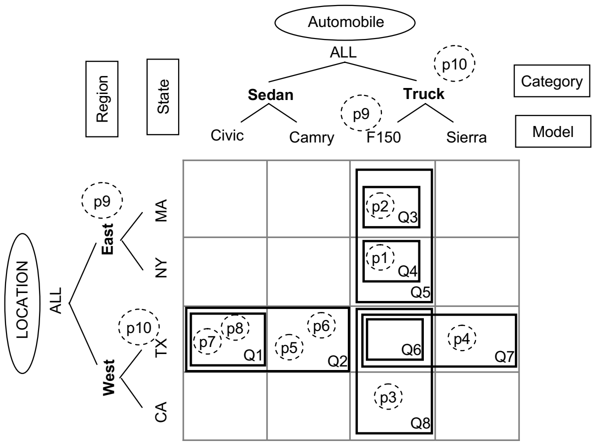


Figure 1: Multidimensional view of the data

For queries whose regions overlap only precise facts, e.g., Q1 and Q2, the set of relevant facts is clear. For other queries, e.g., Q5, this is trickier. If we use the None option, the result of Q5 is $\mathcal{A}$(p1,p2); the imprecise fact p9 is ignored. If we use the Contains option, the result is $\mathcal{A}$(p1,p2,p9). Which answer is better? Using p9 to answer Q5 seems reasonable since the region for Q5 contains p9, and the result reflects all available data. However, there is a subtle issue with using the Contains option to determine relevant facts. In standard OLAP, the answer for Q5 is the aggregate of answers for Q3 and Q4, which is clearly is not the case now, since Q3 = $\mathcal{A}$(p2) and Q4 = $\mathcal{A}$(p1).

Observing that p9 "overlaps" the cells $c1$=`(F150,NY)` and $c2$=`(F150,MA)`, we may choose to *partially* assign p9 to both cells, a process we refer to as *allocation*. The partial assignment is captured by the weights w_{c1} and w_{c2}, such that $w_{c1} + w_{c2} = 1$, which reflect the effect p9 should have on the aggregate value computed for cells $c1$ and $c2$, respectively. If the Overlaps option is used, then Q3 = $\mathcal{A}$(p2, $w_{c1}*$ p9) and Q4 = $\mathcal{A}$(p1, $w_{c2}*$ p9). Observe the user's "expected" relationship between Q3, Q4, and Q5, which we refer to as *consistency*, is now maintained. In addition to consistency, there is a notion of result quality relative to the quality of the data input to the query, which we refer to as *faithfulness*. For example, the answer computed for Q3 should be of higher quality if p9 were precisely known.

To further illustrate the role of allocation, consider query Q6. If p10 is allocated to all cells in its region then Q6 can be answered. Otherwise, the answer to Q6 is undefined, as in regular OLAP. Although allocation can be accomplished in several ways it is reasonable to expect that allocation is query independent. For example, Q7 and Q8 must be answered using the same allocation for p10.

Consistency and faithfulness are discussed further in Sections 3 and 5. A discussion of the possible-world semantics underlying allocation is presented in Section 4, and allocation algorithms are discussed in Section 6. For clarity of exposition, only the statements of the theoretical claims are included in the main body of the paper. Explanations and proofs can be found in [6].

2.5 Aggregating Uncertain Measures

Consider a query of the type "How likely are brake problems for sedans in TX?" This corresponds to query Q2 where the aggregation is over the uncertain measure 'Brake'. The answer to this query is an aggregation over the pdfs for p5, p6, p7, p8. This notion of aggregating pdfs is closely related to the problem studied in the statistics literature under the name of *opinion pooling* [15]. Informally, the opinion pooling problem is to provide a *consensus* opinion from a set of opinions $\mathbf{\Theta}$. The opinions in $\mathbf{\Theta}$ as well as the consensus opinion are represented as pdfs over a discrete domain O.

Many pooling operators have been studied, and the *linear operator* LinOp is among the most widely used. LinOp($\mathbf{\Theta}$) produces a consensus pdf $\overline{P}$ that is a weighted *linear* combination of the pdfs in $\mathbf{\Theta}$, i.e., $\overline{P}(x) = \sum_{P\in\Theta} w_P \cdot P(x)$, for $x \in O$. Here, the weights are non-negative quantities summing to one. Unless there is some form of prior knowledge, we assume that the weights are uniform, i.e., $w_P = 1/|\mathbf{\Theta}|$, in which case $\overline{P}(x)$ is just the average of the probabilities $P(x)$ for $P \in \Theta$. It is straightforward to compute LinOp using aggregation functions in current OLAP systems.

3 OLAP Requirements

In providing support for OLAP-style queries in the presence of imprecision and uncertainty, we argue that the answers to these queries should meet a reasonable set of requirements that can be considered generalizations of requirements met by queries in standard OLAP systems. We propose two requirements for handling imprecision, namely *consistency* and *faithfulness*, which apply to both numeric and uncertain measures. (Some requirements for handling uncertainty have been proposed in [14].) We use these requirements to argue that only the Overlaps option for handling imprecision results in well-behaved queries in the context of OLAP.

3.1 Consistency

The intuition behind the consistency requirement is that a user expects to see some natural relationships hold between the answers to aggregation queries associated with different (connected) regions in a hierarchy.

Definition 7 (α-consistency)**.** Let $\alpha(x, x_1, x_2, \ldots, x_p)$ be a predicate such that each argument of α takes on values from the range of a fixed aggregation operator $\mathcal{A}$. Consider a collection of queries $Q, Q_1, \ldots, Q_p$ such that (1) the query region of Q is partitioned by the query regions of $Q_1, \ldots, Q_p$, i.e., $\mathrm{reg}(Q) = \cup_i \mathrm{reg}(Q_i)$ and $\mathrm{reg}(Q_i) \cap \mathrm{reg}(Q_j) = \emptyset$ for every $i \neq j$, and (2) each query specifies that $\mathcal{A}$ be applied to the same measure attribute. Let $\hat{q}, \hat{q}_1, \ldots, \hat{q}_m$ denote the associated set of answers on D. We say that an algorithm satisfies α-consistency with respect to $\mathcal{A}$ if $\alpha(\hat{q}, \hat{q}_1, \ldots, \hat{q}_p)$ holds for every database D and for every such collection of queries $Q, Q_1, \ldots, Q_p$. $\square$

This notion of consistency is in the spirit of *summarizability*, introduced in [18, 19], although the specific goals are different. Given the nature of the underlying data, only some aggregation functions are appropriate, or have the behavior the user expects.

3.2 Specific Forms of Consistency

We now define appropriate consistency predicates for the aggregation operators considered in this paper, using the notations given in the definition of α-consistency.

Definition 8 (Sum-consistency)**.** Sum-consistency is defined as $\hat{q} = \sum_i \hat{q}_i$. $\square$

The above is an intuitive notion of consistency for SUM and COUNT. Since SUM is a distributive function, SUM for a query region should equal the value obtained by adding the results of SUM for the query sub-regions that partition the region. All statements for SUM in this paper are applicable to COUNT as well, and will not be explicitly mentioned in the interest of space.

Definition 9 (Boundedness-consistency)**.** For a numeric measure, this consistency predicate is defined as $\min_i\{\hat{q}_i\} \leq \hat{q} \leq \max_i\{\hat{q}_i\}$. For an uncertain measure, the above inequalities should hold for the probabilities associated with all elements in the base domain. Formally, if $\hat{q}_(o)$ is the probability for each element o, then $\min_i\{\hat{q}_i(o)\} \leq \hat{q}(o) \leq \max_i\{\hat{q}_i(o)\}$. $\square$

Boundedness-consistency is intuitively appropriate for any kind of averaging operator for numeric measures and aggregation operator for uncertain measures. In particular, AVERAGE for a query region should be within the bounds of AVERAGE for the query sub-regions that partition the region. In the case of LinOp, the same property should hold element-wise for the associated pdfs.

An important consequence of the various α-consistency properties defined above is that the Contains option is unsuitable for handling imprecision, as shown below:

Theorem 1. *There exists a SUM aggregate query which violates Sum-consistency when the Contains option is used to find relevant imprecise facts in* FIND-RELEVANT.

Similar theorems can be shown for other aggregation operators as well, but we omit them in the interest of space.

3.3 Faithfulness

Starting with a database D, suppose we increase imprecision in D by mapping facts in the database to larger regions. We expect that the answer to any query Q on this new database D' will be different from the original answer. Faithfulness is intended to capture the intuitive property that this difference should be as small as possible. Since an aggregation algorithm only gets to see D' as its input and is not aware of the "original" database D one cannot hope in general to state precise lower and upper bounds for this difference. Our aim instead will be to state weaker properties that characterize this difference, e.g., whether it is monotonic with respect to the amount of imprecision. The following definition is helpful in formalizing faithfulness.

Definition 10 (Measure-similar Databases)**.** We say that two databases D and D' are *measure-similar* if D' is obtained from D by (arbitrarily) modifying (only) the dimension attribute values in each fact r. Let $r' \in D'$ denote the fact obtained by modifying $r \in D$; we say that r *corresponds* to r'. $\square$

Consider a query Q such that every fact region is either completely contained within the query region of Q or completely disjoint from it. In such a situation, it is reasonable to treat the facts as if it were precise with respect to Q since the imprecision in the facts does not cause ambiguity with respect to the query region of Q. The first form of faithfulness formalizes this property, and is illustrated in Figure 2a.

Definition 11 (Basic faithfulness)**.** We say that two measure-similar databases D and D' are *identically precise* with respect to query Q if for every pair of corresponding facts $r \in D$ and $r' \in D'$, either both $\mathrm{reg}(r)$ and $\mathrm{reg}(r')$ are completely contained in $\mathrm{reg}(Q)$ or both are completely are disjoint from $\mathrm{reg}(Q)$. We say that an algorithm satisfies *basic faithfulness* with respect to an aggregation function $\mathcal{A}$ if for every query Q that uses $\mathcal{A}$, the algorithm gives identical answers for every pair of measure-similar databases D and D' that are identically precise with respect to Q. $\square$

Basic faithfulness enables us to argue that the None option of handling imprecision by ignoring all imprecise records is inappropriate, as we intuitively expect:

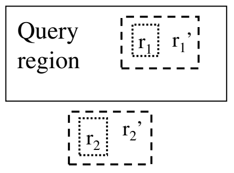
Theorem 2. *SUM, COUNT, AVERAGE and LinOp violate basic faithfulness when the None option is used to handle imprecision.*

Theorems 1 and 2 demonstrate the unsuitability of the Contains and None options for handling imprecision, and we do not consider them further. The remaining option, namely Overlaps, is the focus of our efforts for the rest of the paper, and it raises the challenge of how to handle relevant facts that partially overlap the query region. We tackle this problem in later sections using an allocation-based approach to summarizing the "possible worlds" that may have led to the imprecise dataset.

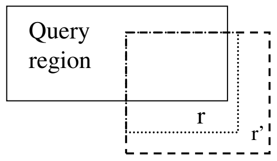
The next form of faithfulness is intended to capture the same intuition as basic faithfulness in the more complex setting of imprecise facts that partially overlap a query. We first define an ordering that compares the amount of imprecision in two databases with respect to a query Q, in order to reason about the answers to Q as the amount of imprecision grows.

Definition 12 (Partial order $\preceq_Q$)**.** Fix a query Q. We say that the relation $I_Q(D, D')$ holds on two measure-similar databases D and D' if all pairs of corresponding facts in D and D' are identical, except for a single pair of facts $r \in D$ and $r' \in D'$ such that $\mathrm{reg}(r')$ is obtained from $\mathrm{reg}(r)$ by adding a cell $c \notin \mathrm{reg}(Q) \cup \mathrm{reg}(r)$. Define the partial order $\preceq_Q$ to be the reflexive, transitive closure of I_Q. □

Figure 2b illustrates the definition of $\preceq_Q$ for a query Q—the amount of imprecision for every fact $r' \in D'$ is larger than that of the corresponding fact $r \in D$ but only in the cells outside the query region. The reason for this restriction is that allowing r' to have a larger projection inside the query region does not necessarily mean that it is less relevant to Q than r (cf. basic faithfulness).



a: No partial overlap



b: Partial order $\preceq_Q$

Figure 2: Two forms of Faithfulness

Definition 13 (β-faithfulness)**.** Let $\beta(x_1, x_2, \ldots, x_p)$ be a predicate such that the value taken by each argument of β belongs to the range of a fixed aggregation operator $\mathcal{A}$.

We say that an algorithm satisfies β-faithfulness with respect to $\mathcal{A}$ if for any query Q compatible with $\mathcal{A}$, and for any set of databases $D_1 \preceq_Q D_2 \preceq_Q \cdots \preceq_Q D_p$, the predicate $\beta(\hat{q}_1, \ldots, \hat{q}_p)$ holds true where $\hat{q}_i$ denotes the answer computed by the algorithm on D_i, i in $1 \ldots p$. □

3.4 Specific Forms of Faithfulness

We now discuss how β-faithfulness applies to the aggregation operations considered in this paper.

SUM: If we consider SUM over non-negative measure values, the intuitive notion of faithfulness is that as the data in a query region becomes imprecise and grows outside the query region, SUM should be non-increasing.

Definition 14 (Sum-faithfulness)**.** Sum-faithfulness is defined as follows: if $D_1 \preceq_Q D_2$, then $\hat{q}_{D_1} \geq \hat{q}_{D_2}$. □

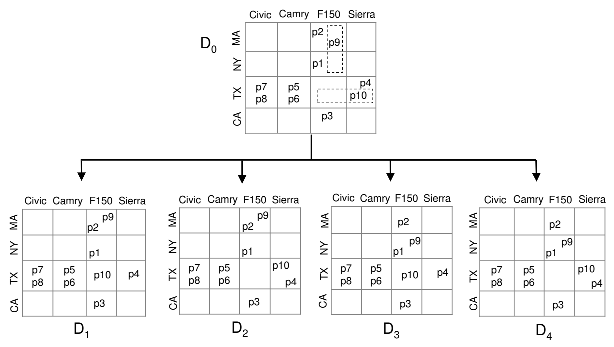


Figure 3: Possible Worlds

AVERAGE and LinOp: It is difficult,unfortunately, to define an appropriate instance of β-faithfulness for AVERAGE and LinOp. Consider how the AVERAGE behave as facts in a query region become more imprecise and grow outside the query region: SUM for the query region diminishes, but the count also decreases. Since both the numerator and denominator are decreasing, the value of AVERAGE could either increase or decrease. The same observation applies to LinOp as well.

4 Possible Worlds

We now describe a *possible-worlds* interpretation of a database D containing imprecise facts, similar to that proposed in [1], as a prelude to defining query semantics when the Overlaps option is used to find relevant facts. Consider an imprecise fact r which maps to a region R of cells. Recall from the discussion following Proposition 1 that each cell in R represents a possible completion of r that eliminates the imprecision in r. Repeating this process for every imprecise fact in D leads to a database D' that contains only precise facts. We call D' a *possible world* for D, and the multiple choices for eliminating imprecision lead to a set of possible worlds for D. We illustrate possible worlds in the following example.

Example 1. Figure 3 shows a multidimensional view of the data in our running example (Figure 1), together with all four possible worlds that can be generated by making the two imprecise facts $p9$ and $p10$ precise. Fact $p9$ can be made precise in two possible ways, placing it in cell `(MA,F150)` or `(NY,F150)`. Similarly, $p10$ can be made precise in two possible ways, placing it in `(TX,F150)` or `(TX,Sierra)`. Different combinations of these $(2 * 2)$ choices lead to the possible worlds $\{D_1, D_2, D_3, D_4\}$. □

We interpret the possible worlds $\{D_1, D_2, \ldots, D_m\}$ as the collection of "true" databases from which the given database D was obtained; the likelihoods of each possible world being the "true" one are not necessarily the same. To capture this likelihood, we associate a non-negative weight w_i with each D_i, normalized so that $\sum_i w_i = 1$. The weights give us flexibility to model the different behaviors that cause imprecision, while the normalization allows for a probabilistic interpretation of the possible worlds.

4.1 Extended Data Model

If there are k imprecise facts in a dataset D, and the region for the i^{th} imprecise fact contains c_i cells, the number of possible worlds is $\prod_{i=1}^{k} c_i$. To tackle the complexity due to this exponential number of possible worlds, we consider each imprecise fact r and assign a probability for its "true" value being c, for each cell c in its region. The assignments for all imprecise facts collectively (and implicitly) associate probabilities (weights) with each possible world, as we explain below.

Definition 15 (Allocation)**.** For a fact r and a cell $c \in \mathrm{reg}(r)$, let $p_{c,r}$ denote the probability that r is completed to c in the underlying "true" world. We call $p_{c,r}$ the *allocation* of fact r to cell c, and require that $\sum_{c \in reg(r)} p_{c,r} = 1$.

Consider the following probabilistic process, starting with a database D containing k imprecise facts: Independently for each imprecise fact r_i, pick a cell c_i with probability p_{c_i,r_i} and modify the dimension attributes in r_i so that the resulting fact belongs to cell c_i. The set of databases that can arise via this process constitute the possible worlds. The weight associated with a possible world D' equals $\prod_{i=1}^{k} p_{c_i,r_i}$.

Any procedure for assigning $p_{c,r}$ is referred to as an *allocation policy*. The result of applying such a policy to a database D is an *allocated database* D^*. The schema of D^* contains all the columns of D plus additional columns to keep track of the cells that have strictly positive allocations. Suppose that fact r in D has a unique identifier denoted by $\mathrm{ID}(r)$. Corresponding to each fact $r \in D$, we create a set of fact(s) $\langle \mathrm{ID}(r), r, c, p_{c,r} \rangle$ in D^* for every $c \in reg(r)$ such that $p_{c,r} > 0$ and $\sum p_{c,r} = 1$. □

Allocation policies are described in detail in Section 6. The size of D^* increases only linearly in the number of imprecise facts. However, since the region of an imprecise fact is exponentially large in the number of dimension attributes which are assigned non-leaf nodes, care must be taken in determining the cells that get positive allocations.

Example 2. For the example in Figure 3, suppose that the probabilities for $p9$ are 0.6 and 0.4 for cells `(MA,F150)` and `(NY,F150)` respectively. Then in D^* we will create two facts corresponding to $p11$—one belonging to `(MA,F150)` with weight 0.6 and another to `(NY,F150)` with weight 0.4 both tagged with the same identifier. Similarly there are 2 facts for $p10$, belonging to `(TX,F150)` and `(TX,Sierra)` with the same id, p10. □

5 Summarizing Possible Worlds

The allocation weights encode a set of possible worlds, $\{D_1, \ldots, D_m\}$ with associated weights $w_1, \ldots, w_m$. The answer to a query Q is a multiset[1] $\{v_1, \ldots, v_m\}$. We are left with the problem of appropriate semantics for summarizing $\{v_1, \ldots v_m\}$.

[1]A multiset because many possible worlds may give the same answer.

Recall that the weights give a probabilistic interpretation of the possible worlds, i.e., database D_i is chosen with probability w_i. We summarize the possible answers $\{v_1, \ldots v_m\}$ by defining a discrete random variable, Z, associated with this distribution.

Definition 16 (Answer variable)**.** Consider the multiset $\{v_1, \ldots v_m\}$ of possible answers to a query Q. We define the *answer variable* Z associated with Q to be a random variable with pdf $\Pr[Z = v_i] = \sum_{j\ s.t.\ v_i = v_j} w_j, i, j \in 1, \ldots, m$. □

The answer to a query can be summarized as the first and the second moments (expected value and variance) of the answer variable Z. Using $E[Z]$ to answer queries is justified by the following theorem:

Theorem 3. *Basic faithfulness is satisfied if answers to queries are computed using the expected value of the answer variable.*

The above approach of summarizing possible worlds for answering aggregation queries, though intuitively appealing, complicates matters because the number of possible worlds grows exponentially in the number of imprecise facts. Allocations can compactly encode this exponentially large set but the challenge now is to summarize without having to explicitly use the allocations to iterate over all possible worlds. We now proceed to design efficient algorithms for summarizing various aggregation operators using the extended data model.

The following notation is useful in the description of the algorithms below. Fix a query Q whose associated region is q. The set of facts that potentially contribute to the answer are those that have positive allocation to q. If $C(r) = \{c \mid p_{c,r} > 0\}$ denotes the set of cells to which fact r has strictly positive allocations, the desired set of facts is given by $\mathcal{R}(Q) = \{r \mid C(r) \cap q \neq \emptyset\}$. We say that $\mathcal{R}(Q)$ is the set of *candidate* facts for the query Q. For any candidate fact r, let $Y_r = Y_{r,Q}$ be the 0-1 indicator random variable for the event that a possible completion of r belongs to q. We have,

$$\Pr[Y_r = 1] = \textstyle\sum_{c \in C(r) \cap q} p_{c,r}$$

Since Y_r is a 0-1 random variable, $\Pr[Y_r = 1] = \mathrm{E}[Y_r]$; the above equation says that $\mathrm{E}[Y_r]$ equals the sum of the allocations of r to the query region of Q. With a slight abuse of notation, we say that $\mathrm{E}[Y_r]$ is the *allocation* of r to the query Q; it is *full* if $\mathrm{E}[Y_r] = 1$ and *partial* otherwise. Finally, note that the independence assumption in our modeling of imprecision implies that the random variables Y_r for the different r's are statistically independent.

We answer query Q in the extended data model in two steps:

Step 1: We identify the set of candidate facts $r \in \mathcal{R}(Q)$ and compute the corresponding allocations to Q. The former is accomplished by using a filter for the query region whereas the latter is accomplished by identifying groups of facts that share the same identifier in the ID column and then summing up the allocations within each group. At the end of this step, we have a set of facts that contains for each fact $r \in \mathcal{R}(Q)$, the allocation of r to Q and the measure value associated with r. Note that this step depends only on the query region q.

Step 2: This step is specialized to the aggregation operator, and two comments are in order. First, we seek to identify the information necessary to compute the summarization while circumventing the enumeration of possible worlds. Second, it is possible in some cases to merge this second step with the first in order to gain further savings, e.g., the expected value of SUM can be computed thus. This extra optimization step will not be discussed further.

5.1 SUM

The random variable corresponding to the answer for a SUM query Q is given by $Z = \sum_{r \in \mathcal{R}(Q)} v_r Y_r$, where v_r is the value of the numerical measure for record r.

Using this expression, we can efficiently compute the expectation and variance for SUM:

Theorem 4. *Expectation and variance can be computed exactly for SUM by a single pass over the set of candidate facts. The expectation of the sum computed from the extended data model satisfies Sum-consistency.*

For SUM, β-faithfulness can be violated if the extended data model was built using arbitrary allocation policies. We define a class of allocation policies for which we can guarantee faithfulness. Such allocation policies will be discussed in Section 6.

Definition 17 (Monotone Allocation Policy)**.** Let D and D' be two similar data sets with the property that the associated regions are identical for every pair of corresponding facts, except for a single pair (r, r'), $r \in D, r' \in D'$ such that $\text{reg}(r') = \text{reg}(r) \cup \{c^*\}$, for some cell c^*. Fix an allocation policy A, and let $p_{c,r}$ (resp. $p'_{c,r}$) denote the resulting allocations in D (resp. D') computed with respect to A. We say that A is a *monotonic* allocation policy if $p_{c,s} \geq p'_{c,s}$ for every fact s and for every cell $c \neq c^*$. $\square$

Monotonicity is a strong but reasonable and intuitive property of allocation policies. When the database has no imprecision, there is a unique possible world with weight 1. But as the amount of imprecision increases, the set of possible worlds will increase as well. Monotone allocation policies restrict the way in which the weights for the larger set of possible worlds are defined. In particular, as a region gets larger, allocations for the old cells are redistributed to the new cells.

Theorem 5. *The expectation of SUM satisfies Sum-faithfulness if the allocation policy used to build the extended data model is monotone.*

5.2 AVERAGE

In this case, the random variable corresponding to the answer is given by $Z = \frac{\sum_{r \in \mathcal{R}(Q)} v_r Y_r}{\sum_{r \in \mathcal{R}(Q)} Y_r}$. Unfortunately, computing even the expectation becomes difficult because of the appearance of Y_r in both the numerator and denominator. As shown in the following theorem, we device a non-trivial algorithm for AVERAGE.

Theorem 6. *Let n and m be the number of partially and completely allocated facts in a query region, respectively. The exact expected value of AVERAGE can be computed in time $O(m + n^3)$, with n passes over the set of candidate facts.*

While the above algorithm is feasible, the cost of computing the exact AVERAGE is high if the number of partially allocated facts for Q is high. To address this, the following theorem shows that we can efficiently compute an approximation for the AVERAGE, given by $Z = \frac{E[\sum_{r \in \mathcal{R}(Q)} v_r Y_r]}{E[\sum_{r \in \mathcal{R}(Q)} Y_r]}$

Theorem 7. *An approximate estimate for AVERAGE can be computed in time $O(m+n)$ using a single pass over the set of candidate facts. The relative error of the estimate is negligible when $n \ll m$.*

The assumption of $n \ll m$ in the theorem above is reasonable for most databases since we expect that the fraction of facts with missing values that contribute to any query will be small.

We now compare our two solutions for AVERAGE, namely the exact and the approximate estimate in terms of the requirements. First, we can show that:

Theorem 8. *The expectation of the AVERAGE computed from the extended data model satisfies basic faithfulness but violates Boundedness-consistency.*

On the other hand:

Theorem 9. *The approximate estimate for AVERAGE defined above satisfies Boundedness-consistency and basic faithfulness.*

The above theorems show the tradeoff between being accurate in answering queries and being consistent. Given the efficiency aspects and the small relative error (under reasonable conditions) for the approximate estimate, we propose using this estimate for answering queries.

5.3 Uncertain Measures

In Section 2.5 we proposed LinOP as a reasonable aggregation operator for uncertain measures. We now address the issue of summarizing LinOp over the possible worlds. One approach is to compute LinOp over all the facts in all the worlds simultaneously, where the facts in a world D_i are weighted by the probability of that world w_i. This is somewhat analogous to the approximate estimate for AVERAGE described above.[2]

Definition 18 (AggLinOp)**.** Let $D_1, D_2, \ldots D_m$ be the possible worlds with weights $w_1, w_2, \ldots w_m$ respectively. Fix a query Q, and let $W(r)$ denote the set of i's such that the cell to which r is mapped in D_i belongs to $\mathrm{reg}(Q)$. AggLinOp is defined as

$$\frac{\sum_{r \in \mathcal{R}(Q)} \sum_{i \in W(r)} v_r w_i}{\sum_{r \in \mathcal{R}(Q)} \sum_{i \in W(r)} w_i},$$

where the vector v_r represent the measure pdf of r. $\square$

Similar to the approximate estimate for AVERAGE, AggLinOp can be computed efficiently, and satisfies similar kinds of requirements.

Theorem 10. *AggLinOp can be computed in a single pass over the set of candidate facts, and satisfies Boundedness-consistency and basic faithfulness.*

6 Allocation Policies

In the previous section, we designed efficient algorithms for various aggregation operators in the extended data model, and proved several consistency and faithfulness properties. We now turn to the task of building the extended data model from the imprecise data via appropriate allocation policies; i.e., design algorithms to obtain $p_{c,r}$ for every imprecise fact r and every cell $c \in \mathrm{reg}(r)$.

As before let $A_1, A_2, \ldots, A_k$ denote the dimension attributes. For any fact r, recall from Proposition 1 that $\mathrm{reg}(r)$ equals some k-dimensional hyper-rectangle $C_1 \times C_2 \times \ldots C_k$ of cells, where each C_i is a subset of the leaf nodes in dom(A_i). Each cell $c \in \mathrm{reg}(r)$ is defined by a tuple $(c_1, c_2, \ldots, c_k)$ where $c_i \in C_i$. Therefore, allocating r to the cell c amounts to replacing the i-th attribute value with c_i for every i. The space of allocation policies is very large, and to facilitate the discussion, we categorize allocation policies as dimension-independent, measure-oblivious, or correlation-preserving.

Definition 19 (Dimension-independent Allocation)**.** An allocation policy is said to be *dimension independent* if the following property holds for every fact r. Suppose $\mathrm{reg}(r) = C_1 \times C_2 \times \ldots C_k$. Then, for every i and every $b \in C_i$, there exist values $\gamma_i(b)$ such that (1) $\sum_{b \in C_i} \gamma_i(b) = 1$ and (2) if $c = (c_1, c_2, \ldots, c_k)$, then $p_{c,r} = \prod_i \gamma_i(c_i)$. $\square$

The above definition can be interpreted in probabilistic terms as choosing *independently* for each i, a leaf node $c_i \in C_i$ with probability $\gamma_i(c_i)$. Part (1) in the above definition ensures that γ_i defines a legal probability distribution on C_i. Part (2) says that allocation $p_{c,r}$ equals the probability cell c is chosen by this process. A *uniform allocation* policy is one where each fact r is uniformly allocated to every cell in $\mathrm{reg}(r)$, and is perhaps the simplest of all policies. We can show that:

[2]A summarizing estimate for uncertain measures that is analogous to the exact estimate for AVERAGE can also be defined but is not considered here because it has the same drawbacks.

Theorem 11. *Uniform allocation is a dimension-independent and monotone allocation policy.*

Even though this policy is simple to implement, a drawback is that the size of the extended data model (which depends on the number of cells with non-zero probabilities) becomes prohibitively large when there are imprecise facts with large regions.

Definition 20 (Measure-oblivious Allocation)**.** An allocation policy is said to be *measure-oblivious* if the following holds. Let D be any database and let D' be obtained from D by possibly modifying the measure attribute values in each fact r arbitrarily but keeping the dimension attribute values in r intact. Then, the allocations produced by the policy are identical for corresponding facts in D and D'. $\square$

Strictly speaking uniform allocation is also a measure-oblivious policy. However, in general, policies in this class do not require the dimensions to be independent. An example of such a policy is *count-based allocation*. Here, the data is divided into two groups consisting of precise and imprecise facts. Let N_c denote the number of precise facts that map to cell c. For each imprecise fact r and cell c,

$$p_{c,r} = \frac{N_c}{\sum_{c' \in \mathrm{reg}(r)} N_{c'}}$$

Thus, the allocation of imprecise facts is determined by the distribution of the precise facts in the cells of the multidimensional space.

Theorem 12. *Count-based allocation is a measure-oblivious and monotone allocation policy.*

A potential drawback of count-based allocation is that once the imprecise facts have been allocated, there is a "rich get richer" effect. To understand this, consider a region. Before allocation, this region has a certain distribution of precise facts over the cells of the region. After count-based allocation, it is highly conceivable that this distribution might be significantly different. In some cases it may be desirable to retain the original distribution exhibited by the precise facts. Applying this requirement to the entire multi-dimensional space motivates the introduction of the *correlation-preserving* class of policies.

Definition 21 (Correlation-Preserving Allocation)**.** Let corr() be a correlation function that can be applied to any database consisting only of precise facts. Let $\Delta()$ be a function that can be used to compute the distance between the results of applying corr() to precise databases.

Let A be any allocation policy. For any database D consisting of precise and imprecise facts, let $D_1, D_2, \ldots, D_m$ be the set of possible worlds for D. Let the $p_{c,r}$'s denote the allocations produced by A on D. Recall by definition 15, that the $p_{c,r}$'s define a weight w_i for D_i, $i \in 1 \ldots m$. The quantity $\Delta(\text{corr}(D_0), \sum_i w_i \cdot \text{corr}(D_i))$ is called the *correlation distance* of A with respect to D. We say that an allocation policy A is *correlation-preserving* if for every database D, the correlation distance of A with respect D is the minimum over all policies.

□

By instantiating corr() with the pdf over dimension and measure attributes $(A_1, \ldots, A_k, M)$ and Δ with the Kullback-Leibler divergence[3] D_{KL}, following Definition 21, we can obtain w_i by minimizing $D_{KL}(P_0, \sum_i w_i P_i)$, where $P_i = \text{corr}(D_i)$, $i \in 0 \ldots m$. Unfortunately, this is a difficult optimization problem since there are an exponentially large number of possible worlds.

6.1 Surrogate Objective Function

Let P denote the pdf $\sum_i w_i P_i$ in the above expression $D_{KL}(P_0, \sum_i w_i P_i)$, where the w_i's are determined from the unknown $p_{c,r}$'s. Since P is a pdf, an appropriate direction that is taken in statistical learning is to treat P as a "statistical model" and obtain the parameters of P by maximizing the likelihood of given data D with respect to P. We will later show how to obtain the allocation weights once we have solved for the parameters of P. The advantage of this method is that it also generalizes very well to the case of uncertain measures, which we now proceed to derive below.

Recall that the value for a fixed uncertain measure attribute in fact r is denoted by the vector v_r, where $v_r(o)$ is the probability associated with the base domain element o. If $v_r(o)$ are viewed as empirical distributions induced by a given sample (i.e., defined by frequencies of events in the sample) then uncertain measures are simply summaries of several individual observations for each fact. Consequently, the likelihood function for this case can written as well. After some simple but not obvious algebra, we obtain the following objective function that is equivalent to the likelihood function:

$$\sum_r D_{KL}\left(v_r, \frac{\sum_{c \in \text{reg}(r)} P_c}{|\operatorname{reg}(r)|}\right),$$

where P_c is the measure distribution for cell c (i.e., the pdf over the base domain)

The vast literature on nonlinear optimization [5] provides several algorithms to obtain a solution for the above optimization problem. But our goal is to obtain the allocation weights $p_{c,r}$, which do not appear in this objective function. Fortunately, the mechanics of the Expectation Maximization (EM) algorithm [11] provides an elegant solution. As described below the *dual variables* in the EM algorithm can be naturally associated with the allocation weights thus providing a convenient link back to the *possible world semantics*. Figure 4 presents the EM algorithm for the likelihood function. The details of the fairly standard derivation are omitted in the interest of space.

Repeat until Converged: **E-step**: For all facts r, cells $c \in \text{reg}(r)$, base domain element o

- $Q(c|r,o) := \dfrac{P_c^{[t]}(o)}{\sum_{c' \in reg(r)} P_{c'}^{[t]}(o)}$

M-step: For all cells c, o

- $P_c^{[t+1]}(o) := \dfrac{\sum_{r:c\in\text{reg}(r)} v_r(o) Q(c|r,o)}{\sum_{o'} \sum_{r:c\in\text{reg}(r)} v_r(o') Q(c|r,o')}$

Figure 4: EM method

Consider now the result of the E-step where we obtain $Q(c|r,o)$. At convergence of the algorithm this represents the posterior distribution over the different values of $c \in \text{reg}(r)$. An alternate pleasing interpretation, in our context, is to view them as the *dual variables* (See [21]). In either view, $Q(c|r,o)$, almost meets our requirements for allocations. One complication is the added dependency on the measure domain o. Each fact r now has as many allocation weights as the number of possible values of o. This is inconsistent with our extended data model. However, this can be easily rectified by marginalizing $Q(c|r,o)$ over o resulting in the following expression.

$$p_{c,r} = Q(c|r) := \sum_o \frac{P_c^{[\infty]}(o)}{\sum_{c'} P_{c'}^{[\infty]}(o)} v_r(o) \qquad (1)$$

Allocation policies for numeric measures can also be derived along the lines of the algorithm described above in a straightforward manner and are omitted in the interests of space.

7 Experiments

In this section we evaluate the main contributions of this paper, namely our extensions to OLAP for handling imprecision and uncertainty. To this end we designed and conducted experiments to evaluate both scalability and quality. The scalability experiments targeted the construction and the querying of the extended data model; the quality experiments targeted the performance of different allocation policies under varying characteristics of the data.

7.1 Scalability of the Extended Data Model

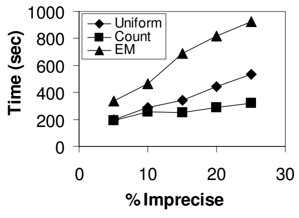
The experiments were conducted on a 2.4GHz Pentium 4 machine with 1 GB physical memory and a single IDE disk. The back-end was a commercial relational database system with buffer pool size set to 100 MB. No materialized views or indices were built on the data.

To provide a controlled environment for evaluation, we used synthetically generated data consisting of 4 dimensions. Experiments using both a numeric measure and an

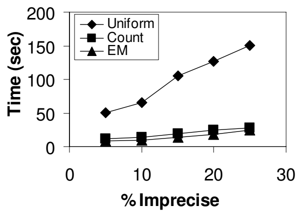
[3]Kullback-Leibler divergence [10] is defined over two distributions P and Q over the same domain as $\sum_x P(x) \log \frac{P(x)}{Q(x)}$.

uncertain measure (over a base domain of size 2) were conducted. All dimensions had hierarchical domains with three levels. For three of these hierarchical domains, the root of the corresponding tree had 5 children; every root child had 10 children each (resulting in 50 leaf nodes); the corresponding branching factors for the remaining dimension was 10 and 10, respectively (100 leaf nodes). Thus, there are 12.5 million cells in the multidimensional space. The initial data consisted of 1 million facts (density=1/12.5 = 8%), each generated by choosing (with uniform probability) a leaf node from the appropriate hierarchy for each dimension. Imprecision was introduced by replacing the leaf node for a dimension with an appropriate parent in the hierarchy. For 50% of the imprecise facts, a second dimension was made imprecise as well (e.g., if 10% of the facts were imprecise, 5% were imprecise in 1 dimension and 5% imprecise in 2 dimensions).

Figure 5a plots the running time for the different allocation policies. Note that they all increase (almost) linearly with respect to the number of imprecise records. The running time has 2 components, one for processing the input data and the other for writing out the facts to the extended data model. For EM the first component is high, since it is an iterative algorithm requiring multiple scans. This explains the reason for longer running time than Uniform and Count which require only a single scan. The larger running time for Uniform with respect to Count is due to the second component. Since the input data density is low, Uniform allocates to many empty cells, so the number of allocated facts created by Uniform is significantly larger than Count and EM. For example, with 25% imprecision, Uniform had 14.5 million facts whereas Count and EM each had 2.3 million facts. This relative difference between Uniform and Count should increase as the input data density decreases.



(a) Allocation Algorithm Running Time



(b) Query Running Time Performance

Figure 5: Performance Results

The second experiment evaluated the performance of standard OLAP point queries using the extended data models created above. For example, SUM can be calculated in SQL using the following template:

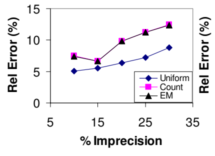
```
SELECT dim-values, SUM (measure * weight),
FROM fact-table, dim-tables
WHERE qualification-list GROUP BY dim-values
```

To understand runtime behavior we randomly generated a total of 25 queries by choosing a random level and node for each dimension. Figure 5b shows the average query running time for SUM. Since the runtime behavior of LinOp and AVERAGE (approximate estimate) were similar, they are omitted in the interests of space. In general the running time was dominated by the I/O cost for scanning the extended data model. As seen above, this is much higher for Uniform than for Count or EM.

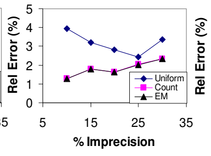
7.2 Quality of the Allocation Policies

These experiments evaluate how data characteristics affect the behavior of our proposed allocation policies. If all facts are precise, dependencies between dimensions are perfectly encoded in the cell counts. As facts become imprecise, a portion of this correlation information is lost. The strength of this encoding against such loss can be measured as the expected number of records in each non-empty cell. We define this new notion of density as *pseudo-density*. Intuitively, high pseudo-density ensures no empty cells are created as records become imprecise. The other characteristic that we chose to examine is measure correlation, which captures the effect of dimension values on the measure value.

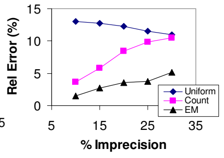
We used synthetically generated data consisting of 2 dimensions with 128 leafs in each dimension (128x128 grid) and a single uncertain measure over the base domain $\{Yes, No\}$. We start by generating a precise data set with the desired pseudo-density and measure correlation. Of the total grid, only 1 out of 8 cells have data records (i.e., regular density is fixed at 12.5%). We then select at random a percentage of records to make imprecise (between 10 - 30%). A record is made imprecise by extending it horizontally over 64 cells. From the resulting imprecise dataset, extended data models are created using different allocation policies (Uniform, Count, EM). For each extended data model, we compute the LinOp aggregate at each cell in the grid. If a cell is empty, then we assign a uniform distribution over the uncertain measure domain (e.g., empty cells are assigned the value (.5, .5) in this case). The quality metric is the average absolute difference for the results as compared to the original precise data set.



(a) Low Pseudo Density Dataset



(b) High Pseudo Density Dataset



(c) Measure Correlated Dataset

Figure 6: Results for Quality Experiments

Figure 6a shows the results for an experiment demonstrating the effects of pseudo-density. The data was generated so no correlation exists between the measure and dimensions. The pseudo-density was set to 1 (i.e., each non-empty cell contains a single record). The results show that Uniform allocation policy has a lower relative error compared to Count and EM. The reason for this is the loss of dimension-value correlation information when a record is made imprecise. For example, if a record *r* in cell *c* is made imprecise, *c* becomes empty, since *r* was the only record in that cell. During allocation, Count and EM will not allocate any portion of *r* to *c*. On the other hand, Uniform will allocate some of *r* to *c*, resulting in a better allocation (i.e., one that better reflects the correct answer).

Figure 6b shows the results for a similar experiment, but with a dataset having a pseudo-density of 4. Again, there is no correlation between the measure and dimensions. Since the pseudo-density is higher, less dimension-value correlation information is lost as more records become imprecise. Thus Count and EM result in better allocations, whereas Uniform suffers since it ignores the available correlation information and allocates to empty cells as well.

Figure 6c shows the results for a data set that has a high correlation between the measure and dimension values. The data was generated so that records in the left half of the grid have measure probability that is high for Yes whereas those in the right half have probability that is larger for No. The pseudo-density was still set to 4. The results show that EM now significantly outperforms both Count and Uniform. This is because EM uses the correlation between the measure and dimensions while performing allocation, whereas Count does not. For example, consider a record *r* in the left half of the grid that is made imprecise to overlap some cells in the right half. Count will allocate *r* to the cells in the right half, whereas EM will allocate *r* only to the cells in the left half since it notices the correlation between the measure value of *r* and cells in the left half.

8 Future Directions

An important aspect of this paper is handling uncertain measures as probability distribution functions (pdfs). The example data in Table 1 provides a conceptual view of this model with a *"pdf"* type column for Brake. Under the assumptions of the model discussed in this paper, adding a new uncertain measure (e.g., Transmission) would result in another column with the same type *"pdf"*. An obvious generalization is to capture the relationships between these two uncertain measures. Consider a query of the type *"How likely are Brake and Transmission problems in Camrys driven in Texas ?"*. This more complicated aggregation query requires additional dependency information between the two uncertain measures and this can be captured as a set of constraints either provided by the user or learned from the data. More generally we believe that the approach initiated here generalizes to handle more general aspects of uncertainty-handling in a DBMS, and we are actively investigating this generalization.

References

[1] S. Abiteboul, P. C. Kanellakis, and G. Grahne. On the Representation and Querying of Sets of Possible Worlds. In *SIGMOD 1987*.

[2] M. Arenas, L. E. Bertossi, and J. Chomicki. Consistent Query Answers in Inconsistent Databases. In *PODS 1999*.

[3] M. Arenas, L. E. Bertossi, J. Chomicki, X. He, V. Raghavan, and J. Spinrad. Scalar Aggregation in Inconsistent Databases. *Theor. Comput. Sci.*, 3(296):405–434, 2003.

[4] D. A. Bell, J. W. Guan, and S. K. Lee. Generalized Union and Project Operations for Pooling Uncertain and Imprecise Information. *Data Knowl. Eng.*, 18(2):89–117, 1996.

[5] D. Bertsekas. *Nonlinear Programming*. Athena Scientific, 1999.

[6] D. Burdick, P. M. Deshpande, T. S. Jayram, R. Ramakrishnan, and S. Vaithyanathan. OLAP Over Uncertain and Imprecise Data. Extended Version, 2005.

[7] R. Cavallo and M. Pittarelli. The Theory of Probabilistic Databases. In *VLDB 1987*.

[8] A. L. P. Chen, J.-S. Chiu, and F. S.-C. Tseng. Evaluating Aggregate Operations over Imprecise Data. *IEEE TKDE*, 8(2):273–284, 1996.

[9] R. Cheng, D. V. Kalashnikov, and S. Prabhakar. Evaluating Probabilistic Queries over Imprecise Data. In *SIGMOD 2003*.

[10] T. Cover and J. Thomas. *Elements of Information Theory*. John Wiley, New York, 1990.

[11] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, B, 1977.

[12] D. Dey and S. Sarkar. PSQL: A Query Language for Probabilistic Relational Data. *Data Knowl. Eng.*, 28(1):107–120, 1998.

[13] H. Garcia-Molina and D. Porter. The Management of Probabilistic Data. *IEEE TKDE*, 4:487–501, 1992.

[14] A. Garg, T. S. Jayram, S. Vaithyanathan, and H. Zhu. Model based opinion pooling. In *8th International Symposium on Artificial Intelligence and Mathematics*, 2004.

[15] C. Genest and J. V. Zidek. Combining Probability Distributions: A Critique and An Annotated Bibliography (avec discussion). *Statistical Science*, 1:114–148, 1986.

[16] J. Kiviniemi, A. Wolski, A. Pesonen, and J. Arminen. Lazy Aggregates for Real-Time OLAP. In *DaWaK 1999*.

[17] L. V. S. Lakshmanan, N. Leone, R. Ross, and V. S. Subrahmanian. ProbView: A Flexible Probabilistic Database System. *ACM TODS*, 22(3):419–469, 1997.

[18] H.-J. Lenz and A. Shoshani. Summarizability in OLAP and Statistical Data Bases. In *SSDBM 1997*.

[19] H.-J. Lenz and B. Thalheim. OLAP Databases and Aggregation Functions. In *SSDBM 2001*.

[20] S. I. McClean, B. W. Scotney, and M. Shapcott. Aggregation of Imprecise and Uncertain Information in Databases. *IEEE TKDE*, 13(6):902–912, 2001.

[21] T. Minka. Expectation-maximization as lower bound maximization., 1998.

[22] A. Motro. Accommodating Imprecision in Database Systems: Issues and Solutions. *SIGMOD Record*, 19(4):69–74, 1990.

[23] A. Motro. Sources of Uncertainty, Imprecision and Inconsistency in Information Systems. In *Uncertainty Management in Information Systems*, pages 9–34. 1996.

[24] T. B. Pedersen, C. S. Jensen, and C. E. Dyreson. Supporting Imprecision in Multidimensional Databases Using Granularities. In *SSDBM*, 1999.

[25] R. Ross, V. S. Subrahmanian, and J. Grant. Aggregate Operators in Probabilistic Databases. *J. ACM*, 52(1):54–101, 2005.

[26] E. A. Rundensteiner and L. Bic. Evaluating Aggregates in Possibilistic Relational Databases. *Data Knowl. Eng.*, 7(3):239–267, 1992.

[27] A. Shoshani. OLAP and Statistical Databases: Similarities and Differences. In *PODS 1997*.

[28] X. Wu and D. Barbará. Learning Missing Values from Summary Constraints. *SIGKDD Explorations*, 4(1):21–30, 2002.

[29] X. Wu and D. Barbará. Modeling and Imputation of Large Incomplete Multidimensional Datasets. In *DaWaK*, 2002.

[30] H. Zhu, S. Vaithyanathan, and M. V. Joshi. Topic Learning from Few Examples. In *PKDD 2003*.