Queries with Arithmetic on Incomplete Databases

Queries with Arithmetic on Incomplete Databases

Marco Console
University of Edinburgh
mconsole@inf.ed.ac.uk

Matthias Hofer
University of Edinburgh
mhofer@inf.ed.ac.uk

Leonid Libkin
University of Edinburgh and
ENS-Paris/PSL
libkin@inf.ed.ac.uk

ABSTRACT

The standard notion of query answering over incomplete database is that of certain answers, guaranteeing correctness regardless of how incomplete data is interpreted. In majority of real-life databases, relations have numerical columns and queries use arithmetic and comparisons. Even though the notion of certain answers still applies, we explain that it becomes much more problematic in situations when missing data occurs in numerical columns.

We propose a new general framework that allows us to assign a measure of certainty to query answers. We test it in the agnostic scenario where we do not have prior information about values of numerical attributes, similarly to the predominant approach in handling incomplete data which assumes that each null can be interpreted as an arbitrary value of the domain. The key technical challenge is the lack of a uniform distribution over the entire domain of numerical attributes, such as real numbers. We overcome this by associating the measure of certainty with the asymptotic behavior of volumes of some subsets of the Euclidean space. We show that this measure is well-defined, and describe approaches to computing and approximating it. While it can be computationally hard, or result in an irrational number, even for simple constraints, we produce polynomial-time randomized approximation schemes with multiplicative guarantees for conjunctive queries, and with additive guarantees for arbitrary first-order queries. We also describe a set of experimental results to confirm the feasibility of this approach.

CCS CONCEPTS

• **Theory of computation** → **Incomplete, inconsistent, and uncertain databases**; *Complexity theory and logic*; **Stochastic approximation**; • **Information systems** → **Incomplete data**;

KEYWORDS

incomplete information, numerical data, missing data, query answering, measure of certainty, first-order queries, approximate query answers, asymptotic behavior

ACM Reference Format:
Marco Console, Matthias Hofer, and Leonid Libkin. 2020. Queries with Arithmetic on Incomplete Databases. In *39th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (PODS'20), June 14–19, 2020, Portland, OR, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3375395.3387666

PODS '20, June 14–19, 2020, Portland, OR, USA

ACM ISBN 978-1-4503-7108-7/20/06…$15.00
https://doi.org/10.1145/3375395.3387666

1 INTRODUCTION

Handling incomplete or uncertain information is one of the central topics in databases, as well as in multiple applications of databases where uncertainty naturally arises. These applications include data integration [25], data exchange [5], ontology-based data access [7, 30], consistent query answering [6], probabilistic databases [31], and others. The standard approach to query answering in all of these is to find answers with some certainty guarantees [19]. Most commonly, one looks for certain answers that are true under all possible interpretations of missing data, although weaker guarantees are possible. In the probabilistic model, for example, we can talk about the probability of a tuple being an answer [31], and can even do this in the absence of explicit distributions [27].

There is one element common to all these approaches. They all start by the standard assumption prevalent in the database literature, which essentially says: assume that there is one domain of values U from which all database elements are drawn. In reality however database columns are typed, and one would need to look hard for a database that does not have numerical columns and arithmetic operations in queries. For example, in the TPC-H benchmark [32], 7 out of 8 tables have numerical attributes, while over 85% of queries use at least order comparisons, and over 70% of queries use standard arithmetic operations such as addition and multiplication. Modeling extra types theoretically is done by viewing relational databases as many-sorted structures. Usually this is a mild assumption for theoretical work, unless there are specific interpreted operations on different types, such as for instance $+, \cdot, <$ on numerical types.

The expressive power and complexity of such languages was extensively studied in the 1990s; see [24] for a summary. For handling incomplete information, however, very little is known. We do know however that the problem becomes considerably harder. For example, under the single-domain assumption, unions of conjunctive queries on databases with nulls can be evaluated efficiently if we only compare values for equality [19], but already in the presence of order comparisons data complexity jumps to coNP, cf. [1]. With more complicated arithmetic, the complexity may jump even higher, all the way to undecidability (we shall give an example later).

At this point we should ask ourselves: what kind of query answers we would actually want in such scenarios. Look at a very simple example: a query $\sigma_{A>B}(R)$ on relation R with attributes A and B and a single tuple $(\bot_1, \bot_2)$ with two nulls. Should the tuple be selected? If we know nothing about $\bot_1$ and $\bot_2$, it seems reasonable to say that with probability $1/2$ the tuple will be in the answer. We now give an example showing the utility of such answers.

Example. A team of sale analysts is asked to predict the effectiveness of the forthcoming sales campaign using data from a database with three relations:

- `Products(id, seg, rrp, dis)` has product ids, their market segment, recommended retail price (`rrp`), and intended discount `dis`;
- `Competition(id, seg, p)` contains information about competing products on the market including their ids, market segment, and offered price `p`;
- `Excluded(id, seg)` has products excluded from the campaign.

As often happens in any business environment, information in these tables may be incomplete while the details of the campaign are being worked out. Furthermore, `Competition` is likely to be populated by an (automated) web extraction algorithm, leading to a high chance of incomplete data in that relation. A sales analyst may want to compile a list of market segments where the company will have a strong competitive advantage, i.e., the price, after discount, of all the products in that segment is below the competition. This can be done with the following query, written in first-order (FO) notation:

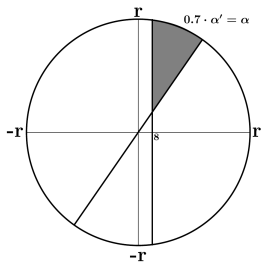
$$q(s) = \forall i, r, d, i'\, p\ \big(\mathsf{P}(i, s, r, d) \wedge \neg \mathsf{E}(i, s) \wedge \mathsf{C}(i', s, p)\big) \rightarrow \\ ((r \cdot d \leq p) \wedge (r, d, p \geq 0))$$

Now consider a database D in which `Competition` has tuple $(c, s, \bot)$, where $\bot$ indicates a null, `Products` has tuples $(id_1, s, 10, 0.8)$ and $(id_2, s, \bot', 0.7)$, where again $\bot'$ indicates a null (here we use the model of *marked* or *labeled* nulls), and `Excluded` has a tuple $(\bot'', s)$ meaning that some product from the segment is excluded from the campaign but we do not yet know which one. We want to estimate how likely it is that s is returned as the answer to q over D. This will happen iff $\bot, \bot'$ are interpreted as numbers α, α' and $\bot''$ as some value u such that $u \neq id_2$ and

$$(\alpha' \geq 0)\ \wedge\ (\alpha \geq 8)\ \wedge\ (0.7 \cdot \alpha' \geq \alpha)\,. \tag{1}$$

Thus, s is not a certain answer to q over D, but it is an answer under conditions (1) and we thus would want to estimate the chance of these conditions being true.

If we know nothing at all about the values that $\bot, \bot', \bot''$ can take, we assume that any value is equally likely. Then, it is rather unlikely that $id_2 = u$ and we can dismiss that constraint. For (1) it seems reasonable then to say: pick α, α' uniformly at random, and compute the probability of (1). The problem is that, of course, we cannot sample from all of $\mathbb{R}$ (or $\mathbb{Z}$ for that matter) uniformly, so we need to restrict the range of numbers to do so. We look then at vectors (α, α') whose length is at most r, and estimate the fraction of the area that constraint (1) covers among all vectors of such length. This is shown in the picture below



The fraction depends on r, but as r increases, one can calculate that it tends to ≈ 0.388 of the positive quadrant, which can be taken as our estimate. Putting a higher discount (e.g., changing 0.7 to 0.5) results in approximately half of the positive quadrant satisfying the constraint, thus increasing our confidence in s being an answer. □

Our goal is to provide a framework in which we can reason about answers in the same manner as in the example. Note that above we did not make any assumptions about probability distributions on data stored in numerical attributes. That is a deliberate assumption, for *this initial study*. We would like the framework to be general enough to accommodate additional information such as distributions or constraints on data. But for now we follow what most of the work on usual incomplete data in relational databases does. There, the standard assumption is that we have a domain of values, and every null can be interpreted as any of the elements of the domain, cf. [19]. We take it further by assuming that we now have a second *numerical* domain, and specifying that some columns take values in that numerical domain. This is similar to stating attribute types as `int` or `float` in `create table` statements. Then we let nulls in numerical columns be interpreted as arbitrary elements of the numerical domain.

Before outlining our contributions, we remark that dealing with missing numerical data is common in data analysis, and typically is dealt with via imputation, i.e., finding suitable replacements for missing data, cf. [14]. There are many variations of the approach that go from simple discarding of tuples with missing values to the inference of missing values through complex statistical models. Crucially though once the imputation is done, the data set is treated as complete, and if it resides in a database, it will be queries using standard techniques for querying complete data. Query answers then will have no way of referring to the fact that some data was initially missing. Our goal is different though: we want to query an incomplete database as given to us, with the full knowledge that it is incomplete, and provide the user with the additional information about confidence levels for potential query answers. Moreover, we want to do it even in the case when there is no prior information about data, such as probability distribution.

Our main contributions are as follows.

(1) We define a model of relational databases with attributes of two distinct types – a base type and a numerical type – with nulls that may occur as values of attributes of both types. We use, as a query language, an extension of relational calculus (FO) with arithmetic operations and order comparisons. We define a measure $\mu(q, D, \bar{a})$ of certainty of a tuple $\bar{a}$ as an answer to query q on database D. It takes values in the interval $[0, 1]$. Value 1 means that the answer is almost surely

certain (essentially being a correct answer with probability 1).

(2) The measure μ relies on computing volumes of sets given by arithmetic constraints, and analyzing their asymptotic behavior. Indeed, we cannot have a uniform distribution over the entire infinite numerical domain, so instead we first analyze all numerical constraints under a bound that all values are bounded by some number r, but then, to avoid dependence on an ad hoc bound, we look at the limit behavior as r grows. This approach poses the question whether the measure is well-defined (i.e., all volumes exist, and limit exists). We prove that it is indeed well-defined.

(3) We outline two approaches to computing the measure $\mu(q, D, \bar{a})$: finding the exact value, and finding its approximation. The former could be computationally hard even for conjunctive queries with inequalities, which is not surprising in view of known complexity results. But with more complex constraints (e.g., linear constraints), values of μ could be irrational, so one needs to approximate.

(4) We then look at the common way of finding approximations, namely constructing an FPRAS (fully polynomial-time randomized approximation scheme), and prove that it exists for conjunctive queries with linear constraints but does not exist for FO queries even with order constraints.

(5) We then notice that values of μ are in the interval $[0, 1]$, and thus it is reasonable to look at slightly weaker approximations a of μ so that $|\mu - a| < \epsilon$ for some small ϵ (under FPRAS, the condition is $|a/\mu - 1| < \epsilon$). Then we prove the existence of such a polynomial-time approximation scheme for all FO queries, with the usual arithmetic (addition, multiplication, order comparisons).

(6) We not only prove the theoretical bounds but also conduct an experimental study to confirm the feasibility of the approach. We look at some decision support queries in an example similar to the one shown earlier, and implement the approximation algorithm for finding confidence levels for candidate tuple answers. We show that its performance is adequate, essentially a small number of seconds with ϵ ranging from 0.1 to 0.01.

Organization In Section 2 we define basic notations related to databases with nulls. In Section 3 we present the model of databases with numerical columns, and in Section 4 we define the measure of certainty μ. We prove that it is well-defined in 5. In Section 6 we outline approaches to computing and approximating μ. In Section 7 we present an FPRAS for conjunctive queries with linear constraints, and in Section 8 we give an approximation scheme with additive guarantees for all FO queries. The results of the experimental study are in Section 9, and conclusions are in Section 10. Due to space limitations, detailed proofs are in the appendix.

2 PRELIMINARIES

We now briefly recall the standard model of incomplete information, where incompleteness is represented by *nulls*, and query answering in this model. In this model, elements of the database come either from a domain C, often viewed as the domain of constants, or known values. They may also come from a domain N of nulls, which are viewed as currently unknown values. A database schema is a set of relation names and their arities. In a database D, a k-ary relation R from the schema is interpreted as k-ary relation over $\mathsf{C} \cup \mathsf{N}$, i.e., a finite set $R^D \subseteq (\mathsf{C} \cup \mathsf{N})^k$. Often D is clear from the context, and we then write R in place of R^D. We write $\mathsf{C}(D)$ and $\mathsf{N}(D)$ for the set of all elements of C and N that appear in relations of D.

Incomplete databases are interpreted by means of *valuations*. A *valuation* is a map $v : \mathsf{N}(D) \to \mathsf{C}$ that interprets nulls by constants. Thus, an incomplete database D represents complete (i.e., having no nulls) databases $v(D)$, where v ranges over valuations, and $v(D)$ is the result of replacing each null $\perp \in \mathsf{N}(D)$ with $v(\perp)$.

Suppose q is a n-ary query that, on each complete database D returns $q(D) \subseteq \mathsf{C}(D)^n$. The standard notion of query answering is that of certain answers [19], i.e., answers independent of the valuation of nulls. For a database D that now may have nulls, an n-tuple $\bar{a}$ over $\mathsf{C}(D) \cup \mathsf{N}(D)$ is a *certain answer* if

$$v(\bar{a}) \ \in \ q\big(v(D)\big) \quad \text{for each valuation } v\,.$$

This is the definition from [28]. One typically sees a restricted form of it when $\bar{a}$ has no nulls; then it says $\bar{a} \in q\big(v(D)\big)$, see [19]. We use the more permissive definition of [28], as it allows tuples with nulls to be in the answer. Its advantages are explained, for example, in [26]. As a simple example, if we have a relation R with a single tuple $(1, \perp)$, and a query returning R, then the definition of [19] will return $\emptyset$, forgetting certain information that we have one tuple in R, and its first component is 1. The definition of [28] that we use here will, on the other hand, return $(1, \perp)$.

It is well-known that certainty comes at a computational price: while for conjunctive queries and their unions and some small extensions with restricted negation they can be computed efficiently [15, 16, 19], for first-order queries (or full relational algebra queries) finding certain answers is coNP-hard in data complexity. This cost however can be significantly reduced if we are only interested in *almost certain* answers, using a simple idea from [27].

Assume the domain of constants comes with some enumeration, $\mathsf{C} = \{c_1, c_2, c_3, \ldots\}$. For a valuation $v : \mathsf{N}(D) \to \mathsf{C}$, define

$$\|v\| = \max\{r \mid c_r \text{ is in the range of } v\}\,.$$

Next, for a query q and a tuple $\bar{a}$, define the r-support of a tuple $\bar{a}$ as an answer to q as the set of valuations v with $\|v\| \le r$ such that $v(\bar{a}) \in q\big(v(D)\big)$, i.e.,

$$\mathrm{Supp}_r(q, D, \bar{a}) = \{v \mid v(\bar{a}) \in q\big(v(D)\big) \text{ and } \|v\| \le r\}\,.$$

We then take $\mu_r(D, q, \bar{a})$ to be the probability that a valuation v with $\|v\| \le r$ picked uniformly at random is in $\mathrm{Supp}_r(q, D, \bar{a})$. As there are only $r^{|\mathsf{N}(D)|}$ valuations v with $\|v\| \le r$ on a database D, this is well defined. Finally, to get rid of the dependence on r, one simply looks at the limit as r grows and thus the set of valuations with $\|v\| \le r$ becomes close to the set of all valuations; that is, $\mu(q, D, \bar{a}) = \lim_{r\to\infty} \mu_r(q, D, \bar{a})$.

Then, [27] established the following results:

- The value of $\mu(q, D, \bar{a})$ does not depend on a particular enumeration of C.
- For a large class of queries the value of $\mu(q, D, \bar{a})$ is either 0 or 1. These queries are generic, i.e., commuting with permutations of their domain. All queries in languages such as FO, relational algebra, datalog, etc., are generic.

- $\mu(q, D, \bar{a}) = 1$ iff $\bar{a}$ is returned by the naïve evaluation of q on D, i.e., treating nulls as new distinct constants, different from all the elements of $\mathsf{C}(D)$.

Thus, a straightforward evaluation of a query on a database with nulls results in answers that are almost certainly true. This happens though in the absence of constraints on database entries, such as their types (e.g., numbers), and in the absence of type-specific operations in queries (e.g., addition and multiplication).

3 DATA MODEL AND QUERY LANGUAGES

We now move to a model where different columns in relations can be of different types. This is a fairly standard multi-sorted model where we allow for nulls as entries.

We assume that database columns can be of two types: either a *base type*, denoted by base, which corresponds to the usual one-domain assumption in the database literature, or a *numerical type*, denoted by num. The domain of base type is denoted by $\mathsf{C}_{\mathsf{base}}$ and the domain of the numerical type by $\mathsf{C}_{\mathsf{num}}$. Thus, what we previously viewed as the set of constants C is now a disjoint union of $\mathsf{C}_{\mathsf{base}}$ and $\mathsf{C}_{\mathsf{num}}$. We assume that the domain of the numerical type $\mathsf{C}_{\mathsf{num}}$ is a subset of the real numbers $\mathbb{R}$.

A schema then, instead of simply specifying the arity of each relation, specifies types of columns. We assume that such specifications are of the form $R(\mathsf{base}^k\mathsf{num}^m)$ saying that the first k columns of R are of base type, and the remaining m are of the numerical type. Such columns, as declared in any real-life DDL, can be interspersed, but here we assume, only for the simplicity of the notation, that all base type columns come first.

Each column, whether of base type or numerical, may contain null values. Using the same marked nulls model, we assume two sets of nulls: $\mathsf{N}_{\mathsf{base}} = \{\bot_1, \ldots\}$ of nulls that occur in base type columns, and $\mathsf{N}_{\mathsf{num}} = \{\top_1, \top_2, \ldots\}$ of numerical type nulls. Then a relation of type $R(\mathsf{base}^k\mathsf{num}^m)$ in a database D is interpreted as a finite set

$$R^D \subset (\mathsf{C}_{\mathsf{base}} \cup \mathsf{N}_{\mathsf{base}})^k \times (\mathsf{C}_{\mathsf{num}} \cup \mathsf{N}_{\mathsf{num}})^m .$$

That is, entries of each base type column come from $\mathsf{C}_{\mathsf{base}} \cup \mathsf{N}_{\mathsf{base}}$ and entries of each numerical type column come from $\mathsf{C}_{\mathsf{num}} \cup \mathsf{N}_{\mathsf{num}}$. We use the same notations to denote sets of elements of each kind in a database D: for example, $\mathsf{C}_{\mathsf{base}}(D)$ is the set of all constants of base type that occur in D, and $\mathsf{N}_{\mathsf{num}}(D)$ is the set of all nulls in numerical columns that occur in D.

Query languages. As our basic query language, we consider two-sorted *first-order logic with arithmetic*, $\mathrm{FO}(+, \cdot, <)$. In this logic, every variable is typed (i.e., of base or numerical type). Its terms and formulae are defined inductively as follows.

Terms • a variable of a base type is a base type term;
- a variable of a numerical type is a numerical type term;
- every element $\alpha \in \mathsf{C}_{\mathsf{num}}$ is a numerical type term;
- if t and t' are numerical type terms, then so are $t + t'$ and $t \cdot t'$.

Atomic formulae • for a relation $R(\mathsf{base}^k\mathsf{num}^m)$, if $\bar{x}$ is a k-tuple of base type variables and $\bar{t}$ is an m-tuple of numerical type terms, then $R(\bar{x}, \bar{t})$ is an atomic formula;
- if x and y are variables of the base type, then $x = y$ is an atomic formula;
- if t, t' are numerical type terms, then $t < t'$ and $t = t'$ are atomic formulae.

Formulae are closed under Boolean connectives and quantifiers:
- If φ, ψ are formulae then $\varphi \vee \psi$, $\varphi \wedge \psi$ and $\neg\varphi$ are formulae;
- if φ is a formula then $\exists x\, \varphi$ and $\forall x\, \varphi$ are formulae.

We shall use the standard shortcuts, i.e., $x \leq y$ for $x < y \vee x = y$ or $2x$ and x^2 for $x + x$ and $x \cdot x$, etc. Since atomic formulae are of the form $t = t'$ or $t < t'$, we can assume that operations $-$ and $\div$ are allowed for building terms as well.

The notion of free variables and the semantics are standard. When queries are interpreted over complete databases D, quantifiers range over their domain. That is, $\exists x$ means that a witness is found among elements of $\mathsf{C}_{\mathsf{base}}(D)$ if x is of base type, and among elements of $\mathsf{C}_{\mathsf{num}}(D)$ if x is of numerical type. For example, $\exists y, z\ R(x, y) \wedge S(y, z) \wedge y > z^2$ for relations $R(\mathsf{base}\,\mathsf{num})$ and $S(\mathsf{num}^2)$ finds all x of base type such that there are tuples $(x, y) \in R$ and $(y, z) \in S$ with $y > z^2$.

Sublanguages. If we restrict the use of arithmetic, we explicitly indicate so; for example, $\mathrm{FO}(+, <)$ means that arithmetic terms are linear functions, built with $+$, and comparisons $t < t'$ are allowed. Notice that with $+$, we also have subtraction: for example, $t_1 - t_2 < t_3$ is $t_1 < t_2 + t_3$.

We shall refer to the $\exists, \wedge$-fragment of the language as *conjunctive queries* and write CQ( ) listing allowed operations in parentheses. For example, CQ($<$) refers to conjunctive queries extended with order comparisons.

4 QUERY ANSWERS: MEASURE OF CERTAINTY

We now describe the approach to query answering in the presence of arithmetic and nulls. As the first step, we need to provide the semantics of incomplete databases over two types. Recall that under the single-type assumption, the semantics was given by valuations, which are functions from nulls to constants C. Now that the set of constants is two-sorted, such valuations on an incomplete database D are pairs $v = (v_{\mathsf{base}}, v_{\mathsf{num}})$ where $v_{\mathsf{base}} : \mathsf{N}_{\mathsf{base}}(D) \to \mathsf{C}_{\mathsf{base}}$ and $v_{\mathsf{num}} : \mathsf{N}_{\mathsf{num}}(D) \to \mathsf{C}_{\mathsf{num}}$. The database $v(D)$ is, as before, obtained by substituting each null $\bot \in \mathsf{N}_{\mathsf{base}}(D)$ with $v_{\mathsf{base}}(\bot)$ and each null $\top \in \mathsf{N}_{\mathsf{num}}(D)$ with $v_{\mathsf{num}}(\top)$. Likewise, when we have a tuple $\bar{a}$, we write $v(\bar{a})$ to denote a tuple obtained from $\bar{a}$ by replacing base type nulls according to v_{base}, numerical type nulls according to v_{num}, and leaving constants intact. For example, $v(\bot, \top, 2) = (v_{\mathsf{base}}(\bot), v_{\mathsf{num}}(\top), 2)$.

Of course with this, we could define certain answers as before: a tuple $\bar{a}$ is a certain answer to query q if $v(\bar{a}) \in q\big(v(D)\big)$ for each valuation v. However, it is well known that even adding simple order comparisons $<$ already leads to much higher complexity of query answering. For example, finding certain answers to conjunctive queries and their unions in the single-type scenario is well-known to be the same as query evaluation [19], i.e., AC^0 in data complexity and NP in combined complexity. However, with order comparisons, the complexity of the problem jumps coNP in data complexity and Π_2^p in combined complexity [1, 21, 35]. Order comparisons lead to further complications when considered in typical scenarios

involving certain answers such as integrating and exchanging data [3, 4].

When arithmetic operations are added, the situation is even worse; in fact we have the following easy observation.

Proposition 4.1. *If $\mathsf{C}_{\mathsf{num}}$ is $\mathbb{Z}$, there is a $\mathrm{CQ}(+,\cdot,<)$ query q such that computing certain answers to q is undecidable, even on a database of a fixed schema, with a single relation consisting of a single tuple.*

proof sketch. Indeed, consider a polynomial $p \in \mathbb{Z}[x_1,\ldots,x_k]$, a relation $R(\mathsf{num}^k)$ with a single tuple $(\top_1,\ldots,\top_k)$, and q given by $\exists \bar{x}\, R(\bar{x}) \wedge p^2 > 0$. Then, a certain answer to this query is true iff p has no integer roots. Since it is know that the problem of finding solutions to Diophantine equations is undecidable even for a fixed number of variable (in fact $k = 13$ [29]), the result follows. □

Thus, it is even more justified in the case of numerical constraints to pass from absolute certainty to measures of certainty, as was done in [27]. This is what we do next.

Measure of certainty. Now consider a query $q(\bar{x},\bar{y})$, where $\bar{x}$ are of base type, $\bar{y}$ are of numerical type, and let D be a database. Let $\bar{a}$ be a tuple of base type constants and nulls from D of the same arity as $\bar{x}$, and let $\bar{s}$ be a tuple of numerical type constants and nulls from D of the same arity as $\bar{y}$. Our goal is to define a measure $\mu(q,D,(\bar{a},\bar{s}))$ of the likelihood of a tuple $(\bar{a},\bar{s})$ being an answer to q on D.

Given a database D, let its base type nulls be $\mathsf{N}_{\mathsf{base}}(D) = (\bot_1,\ldots,\bot_m)$ and its numerical type nulls be $\mathsf{N}_{\mathsf{num}}(D) = (\top_1,\ldots,\top_k)$. Then, for a valuation $v = (v_{\mathsf{base}}, v_{\mathsf{num}})$, we have

$$\begin{array}{lcl} v(\mathsf{N}_{\mathsf{base}}(D)) & \in & \mathsf{C}_{\mathsf{base}}^m \\ v(\mathsf{N}_{\mathsf{num}}(D)) & \in & \mathsf{C}_{\mathsf{num}}^k \quad \subseteq \mathbb{R}^k \end{array}$$

since we assume always that $\mathsf{C}_{\mathsf{num}} \subseteq \mathbb{R}$.

As we did in the previous section, assume that we have an enumeration of base type constants $\mathsf{C}_{\mathsf{base}}$ as $\{c_1, c_2, \ldots\}$ (we shall see soon that the choice of a particular enumeration does not affect the measure we define). When we had only one non-numerical domain we looked at the proportion of valuations v with $\|v\| \le r$ that witness $v(\bar{a}) \in q\big(v(D)\big)$, and then analyzed its asymptotic behavior. Here we do essentially the same. However, notice that $v(\mathsf{N}_{\mathsf{num}}(D))$ is now a vector in $\mathbb{R}^k$, and thus there are infinitely many valuations v with $\|v(\mathsf{N}_{\mathsf{num}}(D))\| \le r$, where $\|\cdot\|$ refers to the usual (Euclidean) norm in $\mathbb{R}^k$. Thus, to measure the number of valuations, instead of cardinality we now look at their *volume*, denoted by Vol.

Thus, we mimic the definition from the previous section. Define

$$\mathsf{Supp}_r(q,D,(\bar{a},\bar{s})) = \left\{ v \;\middle|\; \begin{array}{l} v(\bar{a}) \in q\big(v(D)\big) \\ \|v_{\mathsf{base}}\| \le r \\ \|v_{\mathsf{num}}\| \le r \end{array} \right\},$$

where of course $v = (v_{\mathsf{base}}, v_{\mathsf{num}})$, and where we write $\|v_{\mathsf{base}}\|$ for $\|v_{\mathsf{base}}(\mathsf{N}_{\mathsf{base}}(D))\|$ and likewise for $\|v_{\mathsf{num}}\|$ when D is clear from the context. Next, the idea is essentially the same as before: define the measure of certainty as the probability that a valuation picked uniformly at random belongs to $\mathsf{Supp}_r(q,D,(\bar{a},\bar{s}))$. There is a problems though: the set of valuations v_{base} is discrete and there are finitely many valuations v_{base} with $\|v_{\mathsf{base}}\| \le r$, while for valuations v_{num} we need to measure their volume. We thus need to combine the two in one measure somehow.

To do this, we relativize $\mathsf{Supp}_r(q,D,(\bar{a},\bar{s}))$ for each fixed valuation v_{base} on base-type nulls. That is, we define $\mathsf{Supp}_r(q,D,(\bar{a},\bar{s}) \mid v_{\mathsf{base}})$ as

$$\{v_{\mathsf{num}} \mid (v_{\mathsf{base}}, v_{\mathsf{num}}) \in \mathsf{Supp}_r(q,D,(\bar{a},\bar{s}))\}\,.$$

Since we associate v_{num} with $v_{\mathsf{num}}(\mathsf{N}_{\mathsf{num}}(D)) \subseteq \mathbb{R}^k$, we have $\mathsf{Supp}_r(q,D,(\bar{a},\bar{s}) \mid v_{\mathsf{base}}) \subseteq \mathbb{R}^k$ and thus, if it is measurable, we use its volume as its measure. Then we define

$$\mathsf{Vol}_r(q,D,(\bar{a},\bar{s})) \;=\; \sum_{\|v_{\mathsf{base}}\| \le r} \mathsf{Vol}\big(\mathsf{Supp}_r(q,D,(\bar{a},\bar{s}) \mid v_{\mathsf{base}})\big)$$

as the combined measure of $\mathsf{Supp}_r(q,D,(\bar{a},\bar{s}))$ taking into account the discrete nature of valuations on base type and continuous measure of valuations on the numerical type.

Of course as r grows, so can $\mathsf{Vol}_r(q,D,(\bar{a},\bar{s}))$, since vectors $v_{\mathsf{num}}(\mathsf{N}_{\mathsf{num}}(D))$ come from B_r^k, the k-dimensional ball of radius r. So we normalize this measure, to make it a number between 0 and 1 that determines the likelihood of $(\bar{a},\bar{s})$ being an answer. For that, we divide by the largest possible value of $\mathsf{Vol}_r(q,D,(\bar{a},\bar{s}))$. We have at most $\lfloor r \rfloor^m$ valuations v_{base} with $\|v_{\mathsf{base}}\| \le r$ (we take $\lfloor r \rfloor$ if r is not an integer), and for each of them $\mathsf{Supp}_r(q,D,(\bar{a},\bar{s}) \mid v_{\mathsf{base}}) \subseteq B_r^k$ and thus its volume is bounded by $\mathsf{Vol}(B_r^k)$. Summing up, the maximum value of $\mathsf{Vol}_r(q,D,(\bar{a},\bar{s}))$ is $\mathsf{Vol}(B_r^k) \cdot \lfloor r \rfloor^m$ and thus we define the likelihood as

$$\mu_r(q,D,(\bar{a},\bar{s})) \;=\; \frac{\mathsf{Vol}_r(q,D,(\bar{a},\bar{s}))}{\mathsf{Vol}(B_r^k) \cdot \lfloor r \rfloor^m} \;\in\; [0,1] \tag{2}$$

to be this normalized value.

Finally, as before, we look at the asymptotic behavior of this measure:

$$\mu(q,D,(\bar{a},\bar{s})) \;=\; \lim_{r\to\infty} \mu_r(q,D,(\bar{a},\bar{s}))\,. \tag{3}$$

Convention For Boolean queries q, we shall write $\mu(q,D)$ instead of the more formal $\mu(q,D,())$, i.e., omitting the empty tuple of arguments.

Remark What if there are no numerical variables? In this case $\mathsf{Supp}_r(q,D,(\bar{a},\bar{s}) \mid v_{\mathsf{base}})$ is a subset of $\mathbb{R}^0$, i.e., just a point. If we assume that $\mathsf{Vol}(\mathbb{R}^0) = 1$, then in this case $\mu(q,D,\bar{a})$ can be easily seen to coincide with the measure defined in [27] and presented in the previous section. Thus we indeed provide a proper generalization of the framework of measuring certainty of answers to the case of numerical domains.

5 THE MEASURE IS WELL DEFINED

We provided a formal definition of the measure of certainty $\mu(q,D,(\bar{a},\bar{s}))$ as a number in the interval $[0,1]$, indicating how likely a tuple $(\bar{a},\bar{s})$, consisting of elements of base and numerical types, is to be an answer to q on an incomplete database D. The definition however took for granted the existence of certain quantities, and thus we need to prove that μ is actually well defined. Specifically, there are two elements of the definition of μ where well-definedness needs to be formally proved.

- We defined measure μ using volumes of sets $\mathsf{Supp}_r(q,D,(\bar{a},\bar{s}) \mid v_{\mathsf{base}}) \subseteq \mathbb{R}^k$. But even though $\mathsf{Supp}_r(q,D,(\bar{a},\bar{s}) \mid v_{\mathsf{base}})$ is a bounded set, as a subset of B_r^k, a priori it does not mean that it is measurable and the volume is defined.

- We then defined μ as the limit of some ratio (see equations (2) and (3)) as r grows. However, it is not clear a priori that the limit always exists.

Our goal now is to prove that μ is well-defined for all queries in FO$(+,\cdot,<)$. In the process of showing this, we also reduce the problem of computing $\mu(q,D,(\bar{a},\bar{s}))$ to the problem of analyzing asymptotic behavior of formulae in the first-order theory of the reals. This connection will be essential for obtaining complexity results.

The existence of volumes. For this, we need to prove that sets $\mathsf{Supp}_r(q,D,(\bar{a},\bar{s}) \mid v_{\mathsf{base}}) \subseteq \mathbb{R}^k$ are Lebesgue-measurable.

Proposition 5.1. *For every query $q \in$ FO$(+,\cdot,<)$, database D, tuples $\bar{a}$ and $\bar{s}$, and valuation v_{base} of base type nulls in D, the set $\mathsf{Supp}_r(q,D,(\bar{a},\bar{s}) \mid v_{\mathsf{base}})$ is Lebesgue-measurable, i.e., its volume exits.*

proof sketch. To see this, simply put in the definition of each relation R in $v_{\mathsf{base}}(D)$ (as an explicit disjunction of tuples) in q to see that $\mathsf{Supp}_r(q,D,(\bar{a},\bar{s}) \mid v_{\mathsf{base}})$ is definable in FO theory of the real field $\langle \mathbb{R},+,\cdot,<\rangle$. The latter, by cell decomposition, is a finite union of cells, each of which is open and thus measurable, cf. [34]. □

The existence of limits. We prove this in two steps. First, we restate the definition of μ by fully eliminating its dependence on valuations over the base type. Then we show how to eliminate the database and reduce the question to the existence of limits of volumes of some definable sets over the real field $\langle \mathbb{R},+,\cdot,<\rangle$, which is actually known.

Eliminating base type. We now show that to compute the measure, we can completely disregard valuations v_{base}. Recall that when there is no numerical type, $\mu(q,D,\bar{a}) \in \{0,1\}$, and $\mu(q,D,\bar{a}) = 1$ iff $\bar{a}$ is returned by the naive evaluation of q on D. This observation makes it possible to eliminate base type valuations from the consideration of the asymptotic behavior of μ_r.

For q, D, and a tuple $\bar{a}$, we say that a valuation v_{base} is *bijective* (with respect to $q,D,\bar{a}$) if v_{base} is a bijection and its range is disjoint from $\mathsf{C}_{\mathsf{base}}(D)$. Then it easy to see that for sufficiently large r and two bijective valuations v_{base} and v'_{base} we have

$$\begin{aligned} & \mathsf{Vol}\big(\mathsf{Supp}_r(q,D,(\bar{a},\bar{s}) \mid v_{\mathsf{base}})\big) \\ = \;& \mathsf{Vol}\big(\mathsf{Supp}_r(q,D,(\bar{a},\bar{s}) \mid v'_{\mathsf{base}})\big) \end{aligned}$$

Furthermore, as r grows, almost all (asymptotically) valuations on base type become bijective. This easily leads to the following.

Proposition 5.2. *For a query q, a database D, and tuples $\bar{a},\bar{s}$ of values of free variables of q, let v_{base} be a bijective valuation with respect to $q,D,\bar{a}$. Then*

$$\mu(q,D,(\bar{a},\bar{s})) = \lim_{r\to\infty} \frac{\mathsf{Vol}\big(\mathsf{Supp}_r(q,D,(\bar{a},\bar{s}) \mid v_{\mathsf{base}})\big)}{\mathsf{Vol}(B_r^k)}\,.$$

proof sketch. Using Proposition 5.2 and the observation that

$$\mathsf{Supp}_r(q,D,(\bar{a},\bar{s}) \mid v_{\mathsf{base}}) = \mathsf{Supp}_r(q,v_{\mathsf{base}}(D),(v_{\mathsf{base}}(\bar{a}),\bar{s}))\,,$$

we conclude that for computing $\mu(q,D,(\bar{a},\bar{s}))$ we only need to look at databases D without base type nulls. And once we have such databases, we can further reduce the problem to that for formulae that do not even mention database. □

Eliminating database relations. We now look at databases that have only numerical nulls. Consider such a database D with $\mathsf{N}_{\mathsf{num}}(D) = \{\top_1,\dots,\top_k\}$, an FO$(+,\cdot,<)$ query $q(\bar{x},\bar{y})$, and tuples $\bar{a}$ over $\mathsf{C}_{\mathsf{base}}(D)$ and $\bar{s}$ over $\mathsf{C}_{\mathsf{num}}(D) \cup \mathsf{N}_{\mathsf{num}}(D)$ of the same length as $\bar{x}$ and $\bar{y}$ respectively. Let $\mathbf{R}$ stand for the structure $\langle \mathbb{R},+,\cdot,<\rangle$, i.e., the real closed field.

Proposition 5.3. *Under the assumptions above, there exists, and can be constructed in time polynomial in the size of D, a quantifier-free formula $\varphi_{q,D,\bar{a},\bar{s}}(z_1,\dots,z_k)$ over $\mathbf{R}$ such that $\mathbf{R} \models \varphi_{q,D,\bar{a},\bar{s}}(z_1,\dots,z_k)$ iff $v_{\bar{z}}(\bar{a},\bar{s}) \in q\big(v_{\bar{z}}(D)\big)$, where $v_{\bar{z}}$ assigns z_i to each $\top_i$ for $i \le k$.*

proof sketch. To see this, first associate with each numerical null $\top_i$ a new variable z_i. Then replace all quantifiers $\exists x$ over base type variables by explicit disjunction $\bigvee_{a\in\mathsf{C}_{\mathsf{base}}(D)}$ and quantifiers $\exists y$ over numerical type variables by $\bigvee_{s\in\mathsf{C}_{\mathsf{num}}(D)\cup\mathsf{N}_{\mathsf{num}}(D)}$. In the resulting formula all atomic relational formulae are of the form $R(\bar{c},\bar{u})$, where $\bar{c}$ is a tuple of base type constants, and $\bar{u}$ is a tuple of numerical type constants and nulls of some length p. We then replace such formula by $\bigvee \bigwedge_{j=1}^{p} u_j^\circ = w_j^\circ$ where the disjunction is taken over all tuples $(\bar{c},\bar{w})$ that occur in relation R in D and $c^\circ = c$ for $c \in \mathsf{C}_{\mathsf{num}}$ while $\top_i^\circ = z_i$.

Next, using standard arithmetic, we can assume that each atom $t < t'$ is given as a Boolean combinations of statements $p(\bar{u})\{<,=\}p'(\bar{u})$, where p,p' are polynomials. We then again replace such formulae by $p(u_1^\circ,\dots,u_p^\circ)\{<,=\}p'(u_1^\circ,\dots,u_p^\circ)$, resulting in the desired formula $\varphi(\bar{z})$. Verifying correctness of the translation is routine.

Thus, the set of all valuations v witnessing $v(\bar{a},\bar{s}) \in q\big(v(D)\big)$ is defined by a formula over $\mathbf{R}$. Now consider an arbitrary formula $\varphi(\bar{z})$ with $|\bar{z}| = k$ over $\mathbf{R}$ and define

$$\nu(\varphi) = \lim_{r\to\infty} \frac{\mathsf{Vol}\big(\varphi(\mathbb{R}^k) \cap B_r^k\big)}{\mathsf{Vol}(B_r^k)}\,, \tag{4}$$

where $\varphi(\mathbb{R}^k) = \{\bar{s} \in \mathbb{R}^k \mid \mathbf{R} \models \varphi(\bar{a})\}$. That is, the ratio $\mathsf{Vol}\big(\varphi(\mathbb{R}^k) \cap B_r^k\big)/\mathsf{Vol}(B_r^k)$ shows the proportion of the k-dimensional ball of radius r occupied by vectors satisfying φ, and $\nu(\varphi)$ defines the asymptotic behavior of this ratio. □

Next we use a known fact, namely that $\nu(\varphi)$ exists for every formula φ over $\mathbf{R}$. It was shown in [11], using results of [20, 22], how to definably approximate volumes of sets definable in $\mathbf{R}$. Combining it with previous results, we obtain the main result of this section.

Theorem 5.4. *For an FO$(+,\cdot,<)$ query q, a database D, and tuples $\bar{a},\bar{s}$ of values of free variables of q, the measure $\mu(q,D,(\bar{a},\bar{s}))$ is well-defined and its value is a number in in the interval $[0,1]$. Moreover, one can construct in polynomial time in the size of D a formula $\varphi(\bar{z})$ over $\mathbf{R}$, where $|\bar{z}|$ is the number of numerical nulls in D, such that*

$$\mu(q,D,(\bar{a},\bar{s})) \;=\; \nu(\varphi)\,.$$

We finish this section by returning to the example from the introduction and computing $\mu(q,D,(s))$, where s refers to the only market segment present in the database. Since a bijective valuation v_{base} will not map $\perp''$ to s, we conclude that by Proposition 5.2 and Theorem 5.4 that $\mu(q,D,(s)) = \nu(\varphi)$ where φ is given

by (1). Then an easy calculation shows that for $r > 8$, the ratio $\mathrm{Vol}(\varphi(\mathbb{R}^2) \cap B_r^2)/\mathrm{Vol}(B_r^2)$ is in the interval

$$\left[\frac{(\frac{\pi}{2} - \arctan(\frac{10}{7}))}{2\pi} - \frac{8}{\pi r}, \ \frac{(\frac{\pi}{2} - \arctan(\frac{10}{7}))}{2\pi}\right]$$

which shows that $\nu(\varphi) = (\frac{\pi}{2} - \arctan(\frac{10}{7}))/2\pi \approx 0.097$, or, if we are only interested in the proportion of the positive quadrant satisfying this constraint, it is four times that, i.e., ≈ 0.388 as shown in the Introduction.

6 APPROACHES TO COMPUTING AND APPROXIMATING THE MEASURE

Now that we have the measure of certainty defined, the question is how to compute it. This would be an algorithm that computes a value of a function in the interval $[0, 1]$, rather than a decision problem.

Of course one can attempt to compute the function μ. We shall see though that there are two obstacles along the way. First, the value of μ may be irrational and thus needs to be approximated. Second, computing the value of μ, even if it is rational, may be computationally hard. Ideally, we would like it to be in FP, the class of functions computable in polynomial time. However, we will see that even for simple queries, the problem could be hard for $\mathrm{FP}^{\#P}$, the class of functions computable in polynomial time with calls to a $\#P$ (and thus intractable) oracle.

Starting with the first issue we show the following.

Proposition 6.1. *Consider the Boolean query $q = \exists x, y\, R(x, y) \wedge (x \geq 0) \wedge (y \leq \alpha \cdot x)$ and a database D in which R has a single tuple $R(\top, \top')$. Then $\mu(q, D) \in \mathbb{Q}$ iff $\alpha = 0, \pm 1$.*

proof idea. Indeed, a simple calculation shows that $\mu(q, D) = \arctan(\alpha)/2\pi + 1/2$, and then one can show that it is irrational except for $\alpha = 0$ or $\alpha = \pm 1$. □

In some cases, we can guarantee rationality of $\mu(q, D)$ but then the complexity jumps.

Proposition 6.2. *For queries in $\mathrm{FO}(<)$, the value $\mu(q, D)$ is always rational. However, there are queries for which computing $\mu(q, D)$ for queries from $\mathrm{CQ}(<)$ is $\mathrm{FP}^{\#P}$-hard in data complexity.*

proof sketch. Following [13], computing the rational number $\frac{n}{m} = \mu(q, D)$ is $\mathrm{FP}^{\#P}$-hard if we can find a query q and databases D such that computing the denominator m is done in polynomial time, and computing the numerator n is $\#P$-hard. We can show how to use a $\mathrm{CQ}(<)$ query to compute the number of satisfying assignments to a 3DNF in k variables which is known to be $\#P$-hard [33]; the denominator in this case is 2^k, the number of all assignments, and thus can be computed efficiently in binary. □

These propositions show that it is necessary to look for approximation schemes. Towards that, let us recall some basic definitions, cf. [36]. Given a problem of computing a function $f(\bar{x})$, we say that a randomized algorithm $A(\bar{x}, \epsilon)$ is an FPRAS (fully polynomial-time randomized approximation scheme) for it if A runs in time polynomial in the size of $\bar{x}$ and $1/\epsilon$, and the value $\textsc{output}[A(\bar{x}, \epsilon)]$ it produces satisfies

$$(1 - \epsilon) f(\bar{x}) \ \leq \ \textsc{output}[A(\bar{x}, \epsilon)] \ \leq \ (1 + \epsilon) f(\bar{x}),$$

with probability at least $\frac{3}{4}$. The confidence level $\frac{3}{4}$ can be changed to any arbitrary value $1 - \delta$ for $\delta \in (0, 1)$ by running A polynomially many times over the same input; alternatively δ can be set up as a parameter of the FPRAS and the algorithm A then needs to run in time that is also polynomial in $\log(1/\delta)$.

However, even with order constraints, we cannot get an FPRAS for general FO queries, under a widely believed complexity-theoretic assumption.

Theorem 6.3. *If* NP $\not\subseteq$ BPP, *then there is no FPRAS for computing $\mu(q, D)$ for $\mathrm{FO}(<)$ queries.*

proof sketch. In this case, we produce an FO query $q \in \mathrm{FO}(<)$ and, for each 3CNF formula ψ with n variables, a database D_ψ such that $\mu(q, D_\psi) = \frac{\sharp\psi}{2^n}$, where $\sharp\psi$ is the number of satisfying assignments to ψ. In turn, this proves that ψ is satisfiable if and only if $\mu(q, D_\psi) > 0$, and therefore the associated decision problem is NP-hard. This observation together with a standard complexity theoretic argument allows us to conclude that, if an FPRAS for $\mu(q, D)$ exists, then $NP \not\subseteq BPP$. □

Alternatives. Negative results shown earlier provide us with clear limitations as to what our options could be:

- We could try to find an FPRAS for conjunctive queries, as nothing in our results precludes that; or
- We can try to find a weaker form of approximations for arbitrary $\mathrm{FO}(+, \cdot, <)$ queries.

This is exactly what we do. In the next section we prove the existence of an FPRAS for $\mathrm{CQ}(+, <)$, conjunctive queries with linear constraints. Then, for a weaker form of approximation, we note that the function μ has its values in the interval $[0, 1]$. Thus, unlike FPRAS that provides multiplicative error guarantees, for small values of ϵ we can also settle for a randomized algorithm $A(q, D, (\bar{a}, \bar{s}))$ that returns an answer with *additive error guarantees*, i.e.,

$$|\mu(q, D) - \textsc{output}[A(q, D, (\bar{a}, \bar{s}))]| < \epsilon\,.$$

We provide such an algorithm for all $\mathrm{FO}(+, \cdot, <)$ queries. The algorithm is polynomial in data complexity, i.e., it runs in time polynomial in the size of D and $1/\epsilon$. We will then show experimentally that its performance is feasible not only in theory but also in practice.

7 FRPAS FOR CONJUNCTIVE QUERIES

While, in the general case, computing the likelihood of an answer is hard and cannot be efficiently approximated, in the important restricted case of conjunctive queries with linear constraints approximation is still possible. These queries are essentially joins of relations where join conditions can involve linear (in)equalities such as $\mathtt{rrp} \leq 0.9 \cdot \mathrm{p}$ in the notation of the example in the Introduction, saying that our price is less than 90% of that of the competition.

We show that for this class of queries, $\mathrm{CQ}(+, <)$, we can design an efficient approximation for the measure $\mu(q, D, (\bar{a}, \bar{s}))$ in terms of data complexity. This is based on reduction to formulae in the theory of addition over the reals, along the lines of Proposition 5.3, and then applying an FPRAS for computing the volume of a union of convex bodies based on the existence of some oracles [9].

Given a query $q \in \mathrm{CQ}(+,<)$, we define the function problem $\mu_q(D,(\bar{a},\bar{s}))$ that asks for the values of $\mu(q,D,(\bar{a},\bar{s}))$, with input D and $(\bar{a},\bar{s})$. In other words, this reformulation corresponds to the function problem of computing μ with respect to data complexity.

Theorem 7.1. *Let $q(\bar{x},\bar{y})$ be a query in* $\mathrm{CQ}(+,<)$. *There exists an FPRAS for $\mu_q(D,(\bar{a},\bar{s}))$, with input database D, tuple $(\bar{a},\bar{s})$, and $\epsilon \in (0,1]$.*

That is, there is a randomized algorithm $A_q(D,\epsilon)$ that runs in time polynomial in D and $1/\epsilon$ and returns a value a such that the ratio $a/\mu(q,D,(\bar{a},\bar{s}))$ is in the interval $[1-\epsilon,1+\epsilon]$.

proof sketch. To prove this, we take $q(\bar{x},\bar{y})$, a database D with n numerical nulls, and tuples $\bar{a},\bar{s}$ and use Theorem 5.4 to produce a formula $\varphi(\bar{z})$ with n free variables over $\mathbf{R}$ so that $\mu(q,D,(\bar{a},\bar{s})) = \nu(\varphi)$. To get to the setting where we can define an FPRAS for $\mu_q(D,(\bar{a},\bar{s}))$, we need to analyze this translation in a bit more detail. Recall that first we had to apply a bijective valuation v_{base} to D to get base type nulls out of the way. Since this can be produced in linear time, we can just assume that such a valuation already was applied, and neither D nor $\bar{a}$ have base type nulls.

Next, analyzing the construction of φ given in Proposition 5.3 and given the fact that it is applied to a conjunctive query where existential quantifiers translate into disjunctions and the rest of the formula is a conjunction of predicates, we observe the following about formula φ:

- φ is a formula over $\langle \mathbb{R},+,<\rangle$, i.e., it does not use multiplication;
- φ is of the form $\bigvee_i \bigwedge_j \beta_{ij}(\bar{z})$ where each $\beta_{ij}(\bar{z})$ is an atomic formula over $\langle \mathbb{R},+,<\rangle$, i.e., of the form $\bar{c}\cdot\bar{z}\{<,=\}c'$, where $\bar{c}\cdot\bar{z}$ is the scalar product, $\bar{c}\in\mathbb{R}^n$ and $c'\in\mathbb{R}$.

For such a formula φ, let $\tilde{\varphi}$ denote its *homogenized* version, i.e., the formula obtained from φ by replacing each atomic formula of numerical type $\bar{c}\cdot\bar{z} < c'$ by $\bar{c}\cdot\bar{z} < 0$. Then we know (see [11]) that

$$\nu(\varphi) \;=\; \frac{\mathrm{Vol}(\tilde{\varphi}(\mathbb{R}^n)\cap B_1^n)}{\mathrm{Vol}(B_1^n)}\,.$$

Next, using the special form of φ, we see that $\tilde{\varphi}(\mathbb{R}^n)$ is a finite union of sets $X_1,\ldots,X_m$, corresponding to the disjuncts of φ, and each such set is convex, as the intersection of half-spaces defined by $\bar{c}\cdot\bar{z}<0$. Furthermore, the intersection of each of these sets X_i with B_1^n is convex.

While the problem of computing the volume of a union of convex sets is intractable, in [9] it is shown that it admits an FPRAS under the assumption that each of the sets is given by an individual membership oracle that, for a set $X\in\mathbb{R}^n$ and a point $\bar{a}\in\mathbb{R}^n$, tests whether $\bar{a}\in X$. Since sets are represented as conjunctions of linear constraints and constraints $\|\bar{a}\|\le r$. We have these polynomial-time oracles and can apply the FPRAS of [9] to approximate both $\mathrm{Vol}(\tilde{\varphi}(\mathbb{R}^n)\cap B_1^n)$ and $\mathrm{Vol}(B_1^n)$. It is then an easy observation that we have an FPRAS for their ratio, i.e., $\nu(\varphi)$, and thus for $\mu_q(D,(\bar{a},\bar{s}))$, completing the proof. □

8 FIRST-ORDER QUERIES AND ADDITIVE ERROR APPROXIMATIONS

In Section 7, we proved the existence of an FPRAS for conjunctive queries with linear constraints. This algorithm has two drawbacks: it is limited to conjunctive queries in order to reduce them to a formula of a specific shape, and then it uses that formula in combination with computational geometry algorithms that may not be best suited for implementation inside a DBMS. In contrast, we now prove that, if one settles for an additive error, it is possible to devise an efficient algorithm for all first-order queries with polynomial constraints. This algorithm has the advantage that it is natural to implement, as will be shown in Section 9 that presents an initial experimental evaluation of our approach.

First, we define what it means to have an *additive fully polynomial-time randomized approximation scheme* (AFPRAS) for a problem of computing a function $f(\bar{x})$. An AFPRAS is an algorithm $A(\bar{x},\epsilon)$ that produces a value $\textsc{output}[A(\bar{x},\epsilon)]$ in time polynomial in $\bar{x}$ and $1/\epsilon$ such that

$$f(\bar{x})-\epsilon \;<\; \textsc{output}[A(\bar{x},\epsilon)] \;<\; f(\bar{x})+\epsilon$$

holds with probability at least 3/4.

Once again, we are interested in data complexity, and thus the problem we analyze is that of computing the function $\mu_q(D,(\bar{a},\bar{s}))$, as in the previous section: for a fixed query $q(\bar{x},\bar{y})$, on the input that consists of a database D and values $\bar{a}$ for base type variables $\bar{x}$ and $\bar{s}$ for numerical type variables $\bar{y}$, compute $\mu(q,D,(\bar{a},\bar{s}))$.

Theorem 8.1. *If $q\in \mathrm{FO}(+,\cdot,<)$, then there is an AFPRAS for the function problem $\mu_q(D,(\bar{a},\bar{s}))$.*

In the rest of the section we explain how to construct this algorithm. As in Section 7, we start with $q(\bar{x},\bar{y})$, a database D with n numerical nulls, and tuples $\bar{a},\bar{s}$ and use Theorem 5.4 to produce a formula $\varphi(\bar{z})$ with n free variables over $\mathbf{R}$ so that $\mu(q,D,(\bar{a},\bar{s})) = \nu(\varphi)$. If φ only uses linear constraints, and is in disjunctive normal form, the existence of an AFPRAS follows from results of [11]. Now, however, the query q is in $\mathrm{FO}(+,\cdot,<)$ and thus φ could be a formula over $\mathbf{R}$ of arbitrary shape that also uses multiplication, which complicates the search for an AFPRAS considerably.

The idea of an AFPRAS is to test the asymptotic behavior of φ by testing the truth value of φ far away enough from the origin, along different directions. To formalize this, for a formula $\varphi(\bar{z})$ over $\mathbf{R}$ with n free variables, and for $\bar{a}\in\mathbb{R}^n$, define

$$f_{\varphi,\bar{a}}(k) = \begin{cases} 1, & \text{if } \mathbf{R}\models\varphi(k\cdot\bar{a}) \\ 0, & \text{if } \mathbf{R}\not\models\varphi(k\cdot\bar{a}) \end{cases}$$

where $k\cdot\bar{a}$ is the result of multiplying each component of $\bar{a}$ by k. Intuitively $f_{\varphi,\bar{a}}(k)$ defines the behavior of φ along the direction $\bar{a}$. Functions built this way exhibit a monotonic behavior as the following lemma shows.

Lemma 8.2. *For each formula $\varphi(\bar{z})$ over $\mathbf{R}$ with n free variables, and $\bar{a}\in\mathbb{R}^n$,*

$$\lim_{k\to\infty} f_{\varphi,\bar{a}}(k) \;\in\; \{0,1\}\,.$$

proof sketch. To prove this, notice that the limit, if it exists, is either 0 or 1, as the function $f_{\varphi,\bar{a}}(k)$ assumes only these values. It remains to argue why $\lim_{k\to\infty} f_{\varphi,\bar{a}}(k)$ exists for all $\bar{a}\in\mathbb{R}^n$. Each atomic formula of φ is given as a (multivariate) polynomial constraint. Each of these possesses, into each particular direction represented by $\bar{a}\in\mathbb{R}^n$, finitely many zeros. As a consequence, there is a largest such zero, say k^+, after which all constraints in φ are

either fulfilled or unfulfilled forever. Hence, the value of $f_{\varphi,\bar{a}}(k)$ will be constant for all $k \geq k^+$. □

Using the result presented above, one can prove that the value of $\nu(\varphi)$ depends only on the asymptotic behavior of $f_{\varphi,\bar{a}}$, for $\bar{a}$ taken from the unit ball B_1^n.

Lemma 8.3. *For $\varphi(\bar{z})$ as in Lemma 8.2,*

$$\nu(\varphi) \;=\; \frac{\mathrm{Vol}(\{\bar{a} \in B_1^n \mid \lim_{k\to\infty} f_{\varphi,\bar{a}}(k) = 1\})}{\mathrm{Vol}(B_1^n)}\,.$$

proof sketch. First note that by the proof of Lemma 8.2, the set in the numerator is definable in **R** and thus is measurable and its volume exists. By analogy with the proof of Lemma 8.2, the proportion of directions into which all constraints of φ are eventually always true or false, will be the same regardless of the radius of the ball, and thus to compute it we can use the unit ball and consider the asymptotic behavior of the constraints into each direction. □

The idea of the algorithm then is to sample sufficiently many directions $\bar{a} \in B_1^n$, and test whether $\lim_{k\to\infty} f_{\varphi,\bar{a}}(k) = 1$. To prove that this intuition gives us an efficient algorithm, we first need to prove that testing whether $\lim_{k\to\infty} f_{\varphi,\bar{a}}(k) = 1$ can be done efficiently. This is indeed the case, as the following statement shows.

Lemma 8.4. *Given $\varphi(\bar{z})$ as in previous lemmas, and $\bar{a} \in B_1^n$, checking whether $\lim_{k\to\infty} f_{\varphi,\bar{a}}(k) = 1$ can be done in polynomial time with respect to the size of $\bar{a}$ and φ.*

proof sketch. Indeed, from φ, we first compute the formula $\varphi_{\bar{x}/k\cdot\bar{a}}$ in one free variable k by substituting $k\cdot a_i$ for each variable z_i. Observe that each atomic formula of $\varphi_{\bar{x}/k\cdot\bar{a}}$ is of the form $p(k)\{<,=\}0$, where p is a univariate polynomial. For each such polynomial, to test whether the corresponding constraint is asymptotically fulfilled, we only need to consider the coefficients of the terms of highest degree. □

We finally use all these lemmas to outline the AFPRAS algorithm. Assume an input database D with $|\mathsf{N}_{\mathsf{num}}(D)| = n$, a tuple $(\bar{a}, \bar{s})$ of the appropriate type, and a value $\epsilon \in (0, 1]$. First, using Theorem 5.4 we produce, in polynomial time, a formula $\varphi(\bar{z})$ with n free variables over **R** so that $\mu(q, D, (\bar{a}, \bar{s})) = \nu(\varphi)$. Then, to compute $\nu(\varphi)$ approximately, we sample, uniformly at random, m vectors $\bar{a}_1, \ldots, \bar{a}_m$ from the unit n-dimensional ball (see [8] for a description of such a sampling). Then, we compute the values of

$$A_m(D) \;=\; \sum_{i=1}^{m} \frac{\lim_{k\to\infty} f_{\varphi,\bar{a}_i}(k)}{m}\,.$$

Using the Chernoff bound, we can prove that, if $m \geq \epsilon^{-2}$, then $A_m(D) \in [\mu_q(D, (\bar{a}, \bar{s})) - \epsilon, \mu_q(D, (\bar{a}, \bar{s})) + \epsilon]$ with probability at least $\frac{3}{4}$, which shows that for such m the algorithm is the desired AFPRAS. □

9 EXPERIMENTAL EVALUATION

We have implemented the additive error approximation algorithm from Section 8. Its main goal was not to produce a fully fledged system for generating a confidence level for queries, but rather to make sure that the proposed theoretical scheme, in addition to having nice theoretical properties, is also feasible in practice.

Specifically, we used this implementation to evaluate a set of join queries with numerical conditions that are fairly typical decision support queries. We evaluated the algorithm for various precision levels ranging from 0.1 to 0.01, essentially corresponding to getting one or two decimal digits correctly. The main parameter to measure was the time overhead required to run the algorithm. The key conclusion is that for common decision queries, and even fairly small ϵ (i.e., high confidence level), getting approximate results is feasible at least for medium size databases.

Implementation. The implementation of the approximation algorithm itself is done in Python, using a standard library (Numpy) to perform the random sampling. It works on the output of a query, under naive evaluation, produced by the Postgres database, to evaluate confidence level in tuples it generates. Our implementation follows very closely the theoretical algorithm presented in Section 8. Given a conjunctive query q and a database D, we first use Postgres to construct a compact representation of the formulae $\varphi_{q,D,\bar{a},\bar{s}}$ defined in Section 5. With it in place, we run the Monte-Carlo phase of our algorithm, with only one major difference.

In the theoretical algorithm, given a database D with n different numerical nulls and a query q, we sample a vector $\bar{z}$ uniformly at random from the n-dimensional unit ball and check the asymptotic behavior of q along the direction of $\bar{z}$. To sample $\bar{z}$ uniformly, we use the standard technique of sampling n independent and normally distributed random variables, and then scale the vector obtained this way to fit the unit ball, see [8].

While this technique is efficient in theory, in practice sampling normally distributed random variables requires the use of a relatively slow Python routine. When there are many nulls in the database, this inevitably leads to poor performances. In our implementation, instead of sampling the whole $\bar{z}$, we only sample as many many coordinates of $\bar{z}$ as needed to replace the nulls that affect the result of the input query. As the final scaling does not affect the direction represented by $\bar{a}$, we can safely use this partial vector for our asymptotic analysis. This optimization saves us a considerable amount of calls to the sampling routine and speeds up the computation substantially.

Dataset and Queries. To perform our experiments, we used synthetic data generated for a schema of a sales database resembling the one we used in the introduction. It has the following relations.

- `Products(id, seg, rrp, dis)` has product ids, their market segment (`seg`), recommended retail price (`rrp`), and the intended discount (`dis`).
- `Orders(id, pr, q, dis)` contains information about possible future orders, including the product id (`pr`), the quantity ordered (`q`), and the extra discount applied on the order (`dis`). In this case, the final discount depends on the quantity, and is calculated as `dis/q`.
- `Market(seg, rrp, dis)` stores the recommended retail price (`rrp`) and the forecast discount (`dis`) of the best competing product on the market.

To produce the data, we used DataFiller [10], a tool for the generation of synthetic data. We generated an SQL database of about 200K tuples, with null values. To use this data in our assessment, we replaced each SQL's null with a different string so to obtain the

equivalent of a database with marked nulls. We then loaded the data into Postgres to perform our evaluation.

We used three typical decision support queries in our experiments. They ask for segments of the market where competitive advantage could be achieved, for having much better offers than the market, and for other companies offering unfair discounts. As is common in such decision support queries, we put a `limit` clause to present the analyst with an analyzable sample from the database. Below we give SQL code and brief descriptions of queries.

- Competitive Advantage Query: What are the market segments where the company will have a competitive advantage?

```
SELECT P.seg
FROM Products P, Market M
WHERE P.seg = M.seg AND
      P.rrp * P.dis <= M.rrp * M.dis
LIMIT 25
```

- Never Knowingly Undersold Query: What are the products that will sell for less than half of the best price on the market?

```
SELECT P.id
FROM Products P, Orders O, Market M
WHERE P.seg = M.seg AND
   (P.rrp * P.dis * (O.q/O.dis)
    <= 0.5 * M.id * M.dis)
LIMIT 25
```

- Unfair Discount Query: What are the orders that apply a discount at least 60% higher than the intended campaign discount?

```
SELECT O.id
FROM Products P, Orders O, Market M
WHERE P.id = O.pr AND
   (P.rrp * P.dis O.q) <= (0.5 * M.id * M.dis)
LIMIT 25
```

Experimental Setup and Results. We ran our implementation for each of the above queries, using errors from 0.01 to 0.1 with a step of 0.005. This resulted in 19 different figures for each of the queries. All experiments ran on a machine with an Intel(R) Core(TM) i5-8500 CPU @ 3.00GHz processor, 16GB of RAM, 500GB hard-disk, running Xubuntu 19.04.

The results are shown in Figure 1. Along the x axis, we show error levels ϵ, and along the y axis we show running time of the algorithm. Smaller values of ϵ correspond to higher precision and thus more sampling and longer running times. If we are interested only in the precision up to the first decimal digit ($\epsilon \in (0, 0.1]$), then our implementation, for all of the queries, runs in less than a second. This is a very small overhead that produces valuable information on the likelihood of tuples returned by the naive implementation of a query, and in particular tells the analyst whether results that may be based on incomplete information warrant further investigation. It is also reasonable to expect that such a level of precision would be acceptable. After all, it should not matter that much whether the likelihood of a tuple is 0.8 or 0.7. As the precision increases, and ϵ decreases, the performance degrades as expected. But even in the case when the second decimal digits matters ($\epsilon \in (0, 0.01]$), the overhead time required to perform our analysis stays below 10 seconds for every single query.

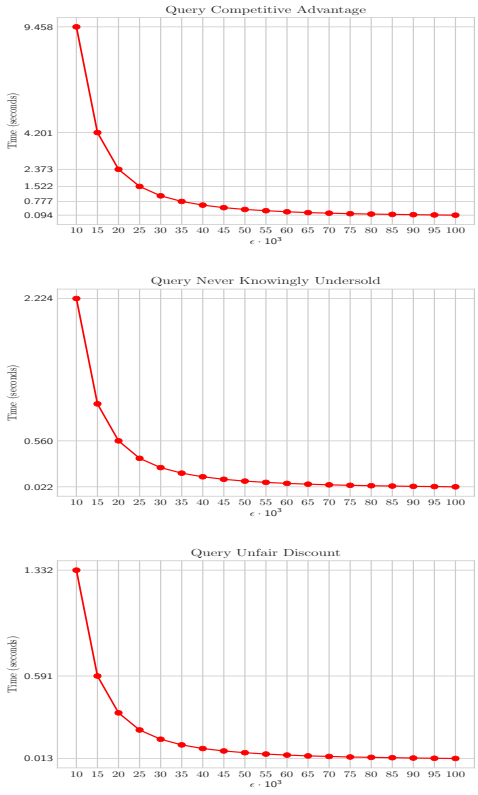


Figure 1: Experimental results

10 FUTURE WORK

The good theoretical bounds as well as promising results of the experimental evaluation suggest pushing this approach further in at least several directions.

The most important one is to move away from the totally agnostic approach where we say that each numerical null can be interpreted by an arbitrary element of the numerical domain. While this was motivated by the usual assumption in the literature on incomplete databases that nulls can denote arbitrary elements, and as such it was appropriate for this initial study, in real life with typed columns additional constraints are possible. Most commonly we have restrictions on ranges of numerical attributes. For example, price is expected to be positive, while discount is expected to be a number in $[0, 1]$. However the model we proposed here is very easily adaptable to such modifications. We can simply add such constraints in both the numerator and denominator of the ratio defining the measure of certainty. It then needs to be seen how the AFPRAS algorithm is to be adjusted. The model is also suitable for adding probability distributions associated with particular columns, which can simply replace uniform distributions over the n-dimensional ball.

So far we measured the likelihoods of answers by computing volumes in $\mathbb{R}^n$. In the case of integer type, we can provide an alternative measure by counting the number of integer lattice points. In general one may expect to obtain similar results as the n-dimensional

generalization of the Gauss circle problem says that the number of integer points inside B_r^n is a good approximation of $\mathrm{Vol}(B_r^n)$, i.e., the difference between such number and $\mathrm{Vol}(B_r^n)$ is of the order $o(\mathrm{Vol}(B_r^n))$, cf. [23]. Extending the measure in the integer case may also offer connections with the recent work on open world probabilistic databases over countable domains [17]. Open world databases with uncountable domains [18] offer another possible direction for further exploration.

As further extensions, one could look at extensions with aggregate queries, where reasoning about certainty is significantly more complicated [2] but ideas related to approximation might perhaps be helpful in the case of numerical answers. Yet another connection worth exploring is with the study of the asymptotic behavior of queries over random databases under the constraint that the expected size of the database remains constant [12], as this is indeed close to the model we have with substituting values for nulls.

Acknowledgment Work partially supported by EPSRC grants M025268, N023056, and S003800. Part of this work was done while the third author was at IRIF, Université de Paris, and DI-ENS, supported by a grant from the Foundation Sciences Mathématiques de Paris.

REFERENCES

[1] Serge Abiteboul and Oliver Duschka. 1998. Complexity of answering queries using materialized views. In *PODS*. 254–263.
[2] Foto Afrati and Phokion Kolaitis. 2008. Answering aggregate queries in data exchange. In *PODS*. 129–138.
[3] Foto Afrati, Chen Li, and Prasenjit Mitra. 2002. Answering Queries Using Views with Arithmetic Comparisons. In *PODS*. 209–220.
[4] Foto Afrati, Chen Li, and Vassia Pavlaki. 2008. Data exchange in the presence of arithmetic comparisons. In *EDBT*. 487–498.
[5] Marcelo Arenas, Pablo Barceló, Leonid Libkin, and Filip Murlak. 2014. *Foundations of Data Exchange*. Cambridge University Press.
[6] Leopoldo Bertossi. 2011. *Database Repairing and Consistent Query Answering*. Morgan&Claypool Publishers.
[7] Meghyn Bienvenu and Magdalena Ortiz. 2015. Ontology-Mediated Query Answering with Data-Tractable Description Logics. In *Reasoning Web*. 218–307.
[8] Avrim Blum, John Hopcroft, and Ravindran Kannan. 2020. *Foundations of Data Science*. CUP.
[9] Karl Bringmann and Tobias Friedrich. 2010. Approximating the volume of unions and intersections of high-dimensional geometric objects. *Comput. Geom.* 43, 6-7 (2010), 601–610.
[10] Fabien Coelho. [n.d.]. DataFiller – generate random data from database schema. https://www.cri.ensmp.fr/people/coelho/datafiller.html.
[11] Marco Console, Matthias Hofer, and Leonid Libkin. 2019. Measuring the Likelihood of Numerical Constraints. In *IJCAI*.
[12] Nilesh N. Dalvi, Gerome Miklau, and Dan Suciu. 2005. Asymptotic Conditional Probabilities for Conjunctive Queries. In *ICDT*. 289–305.
[13] Bettina Fazzinga, Sergio Flesca, and Francesco Parisi. 2015. On the Complexity of Probabilistic Abstract Argumentation Frameworks. *ACM Trans. Comput. Log.* 16, 3 (2015), 22:1–22:39.
[14] Yunjun Gao and Xiaoye Miao. 2018. *Query Processing over Incomplete Databases*. Morgan & Claypool Publishers.
[15] Amélie Gheerbrant and Leonid Libkin. 2015. Certain Answers over Incomplete XML Documents: Extending Tractability Boundary. *Theory Comput. Syst.* 57, 4 (2015), 892–926.
[16] Amélie Gheerbrant, Leonid Libkin, and Cristina Sirangelo. 2014. Naïve evaluation of queries over incomplete databases. *ACM Trans. Database Syst.* 39, 4 (2014), 31:1–31:42.
[17] Martin Grohe and Peter Lindner. 2019. Probabilistic Databases with an Infinite Open-World Assumption. In *PODS*. 17–31.
[18] Martin Grohe and Peter Lindner. 2020. Infinite Probabilistic Databases. In *23rd International Conference on Database Theory (ICDT 2020) (Leibniz International Proceedings in Informatics (LIPIcs))*, Carsten Lutz and Jean Christoph Jung (Eds.), Vol. 155. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 16:1–16:20. https://doi.org/10.4230/LIPIcs.ICDT.2020.16
[19] Tomasz Imielinski and Witold Lipski. 1984. Incomplete information in relational databases. *J. ACM* 31, 4 (1984), 761–791.
[20] Marek Karpinski and Angus Macintyre. 1997. Approximating the Volume of General Pfaffian Bodies. In *Structures in Logic and Computer Science, A Selection of Essays in Honor of Andrzej Ehrenfeucht*. 162–173. https://doi.org/10.1007/3-540-63246-8_10
[21] Anthony C. Klug. 1988. On conjunctive queries containing inequalities. *J. ACM* 35, 1 (1988), 146–160.
[22] Pascal Koiran. 1995. Approximating the volume of definable sets. In *FOCS*. 134–141. https://doi.org/10.1109/SFCS.1995.492470
[23] E. Krätzel. 1988. *Lattice Points*. Kluwer.
[24] Gabriel Kuper, Leonid Libkin, and Jan Paredaens. 2000. *Constraint Databases*. Springer.
[25] Maurizio Lenzerini. 2002. Data integration: a theoretical perspective. In *ACM Symposium on Principles of Database Systems (PODS)*. 233–246.
[26] Leonid Libkin. 2016. SQL's Three-Valued Logic and Certain Answers. *ACM Trans. Database Syst.* 41, 1 (2016), 1:1–1:28.
[27] Leonid Libkin. 2018. Certain Answers Meet Zero-One Laws. In *PODS*. 195–207.
[28] Witold Lipski. 1984. On Relational Algebra with Marked Nulls. In *PODS*. 201–203.
[29] Ju. V. Matijasevic and Julia Robinson. 1975. Reduction of an arbitrary diophantine equation to one in 13 unknowns. *Acta Arith.* 27 (1975), 521–553.
[30] Antonella Poggi, Domenico Lembo, Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Riccardo Rosati. 2008. Linking Data to Ontologies. *J. Data Semantics* 10 (2008), 133–173.
[31] D. Suciu, D. Olteanu, C. Re, and C. Koch. 2011. *Probabilistic Databases*. Morgan&Claypool Publishers.
[32] Transaction Processing Performance Council 2014. *TPC Benchmark™ H Standard Specification*. Transaction Processing Performance Council. Revision 2.17.1.
[33] Leslie G. Valiant. 1979. The Complexity of Enumeration and Reliability Problems. *SIAM J. Comput.* 8, 3 (1979), 410–421. https://doi.org/10.1137/0208032
[34] Lou Van den Dries. 1998. *Tame Topology and o-minimal Structures*. Vol. 248. Cambridge university press.
[35] Ron van der Meyden. 1997. The Complexity of Querying Indefinite Data about Linearly Ordered Domains. *J. Comput. Syst. Sci.* 54, 1 (1997), 113–135.
[36] Vijay V. Vazirani. 2001. *Approximation Algorithms*. Springer. http://www.springer.com/computer/theoretical+computer+science/book/978-3-540-65367-7