# An Operational Approach to Consistent Query Answering*

Marco Calautti, Leonid Libkin, and Andreas Pieris
{mcalautt,libkin,apieris}@inf.ed.ac.uk

School of Informatics, University of Edinburgh

## 1 Introduction

Consistent query answering (CQA) is an elegant idea introduced in the late 1990s by Arenas, Bertossi, and Chomicki [1] that has been extensively studied since. The main premise is that databases are often *inconsistent*, i.e., do not conform to their specifications in the form of integrity constraints. The reason behind this is that data is not perfect and clean: it may come, for instance, from several conflicting sources. Data cleaning attempts to fix this problem but it is not always possible and some inconsistencies remain. In such a case CQA aims to deliver meaningful answers to queries that can still be obtained from inconsistent data.

The key elements of the CQA approach are the notion of *repair* of an inconsistent database $D$, that is, a consistent database whose difference with $D$ is somehow minimal, and the notion of query answering based on *certain answers* (i.e., one looks at answers that are true in all repairs). Since there could be many repairs, finding certain answers is most commonly coNP-hard, even for conjunctive queries [3, 9]. This led to a large body of work on showing dichotomy results; see, e.g., [6, 7], classifying all query answering into tractable and coNP-hard cases as the ultimate goal of the CQA endeavor.

But even obtaining good sufficient conditions for tractability leaves many relevant queries beyond reach of the CQA approach. Thus, the standard approach, while yielding good theoretical results, appears to be a bit of dead end which is reflected by its limited practical applicability [4, 8]. We would like to rectify this. We believe that the ultimate goal of a practically applicable CQA approach should be *efficient approximate query answering* with explicitly stated guarantees. However, in the current state of affairs this goal does not seem to be attainable. Efficient probabilistic algorithms with bounded one-sided or two-sided error are unlikely for CQA: placing it in tractable randomized complexity classes such as RP or BPP would imply that the polynomial hierarchy collapses [5]. For coming up with more refined approximation techniques, the current CQA framework lacks flexibility and finer details related to its key concepts.

Our goal is to replace the current declarative approach to repairs with an *operational* one that explains the process of constructing a repair. As it gives us a finer understanding of why an instance is a repair, it also leads to more refined ways of answering queries, by letting us define *how certain* we are that a tuple should be in the answer. This in turn opens up the possibility of efficient approximate consistent query answering.

* This is a short version of [2].

## 2 Outline of the Operational Approach

The key elements of the new approach to database repairs are:

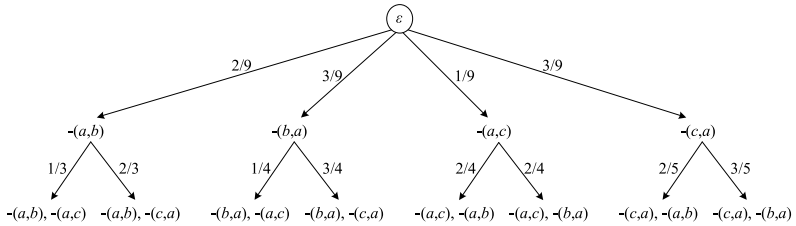
1. the notion of *violation* of a constraint, which simply explains why a constraint is not satisfied by the database;
2. *repairing sequences of operations* on databases, where an operation (e.g., insertion or deletion of tuples) aims at eliminating violations of constraints, while a repairing sequence applies operations to a database until a consistent database is produced;
3. assigning *likelihood* to repairs based on operations used in repairing sequences; and
4. *flexible query answering* based on the likelihood of a tuple appearing in an answer on different repairs.

Let us illustrate the above elements via a simple example. Consider the database

$$D \;=\; \{\text{Pref}(a,b), \text{Pref}(a,c), \text{Pref}(a,d), \text{Pref}(b,a), \text{Pref}(b,d), \text{Pref}(c,a)\},$$

and the set $\Sigma$ that contains a single denial constraint $\text{Pref}(x,y), \text{Pref}(y,x) \rightarrow \bot$, which states that the preference relation over, e.g., products, is irreflexive and anti-symmetric. It is clear that $D$ is inconsistent w.r.t. $\Sigma$. Our goal is to repair $D$ via a repairing sequence of operations, which, in this case, are always deletions. However, during the repairing process, we would like to take into account the fact that some products have more support than other ones. For example, $a$ has more support than $b$ in $D$ since $a$ is preferred more often than $b$. This is achieved by applying the operation $-\text{Pref}(b,a)$, which simply removes the atom $\text{Pref}(b,a)$, with higher probability than $-\text{Pref}(a,b)$ since it is more likely that $a$ is preferred over $b$, and thus we would like to keep $\text{Pref}(a,b)$ with higher probability than removing it.

Our intention described above can be nicely captured via a tree-shaped Markov chain $M$ like the one shown below. The Markov chain $M$ is basically a tree that encodes all the possible repairing sequences that lead to a database that is consistent with $\Sigma$:




For brevity, we omit the predicate Pref in the figure, i.e., instead of writing $-\text{Pref}(a,b)$ we simply write $-(a,b)$. The states of $M$ are repairing sequences with $\varepsilon$ being the empty sequence, which is by definition repairing. The edges are labeled with a probability $p \in [0,1]$, which is simply the probability of moving from one state to another.

Starting from the database $D$, the probability of removing $\text{Pref}(b,a)$ is $3/9$, and the probability of removing $\text{Pref}(a,b)$ is $2/9$. This captures our intention of keeping $\text{Pref}(a,b)$ with higher probability than removing it since $a$ has more support than $b$ in $D$. In fact, these probabilities are not arbitrary but rather are provided by a precise algorithm that extracts them from the data; for details see [2]. Analogously, since $a$ has more

support than $c$ in $D$, the probability of removing $\text{Pref}(c, a)$ is higher than the probability of removing $\text{Pref}(a, c)$. Now, assume that we choose to apply $-\text{Pref}(b, a)$, and thus construct the database $D' = D - \{\text{Pref}(b, a)\}$. The probability of removing $\text{Pref}(c, a)$ is $3/4$, while the probability of removing $\text{Pref}(a, c)$ is $1/4$, which again captures our intention of keeping $\text{Pref}(a, c)$ with higher probability than removing it since $a$ has more support than $c$ in $D'$. Observe that the leaves of $M$ are repairing sequences that lead to a database that satisfies $\Sigma$. These sequences are *complete*, i.e., they cannot be extended further. Since in a Markov chain the probabilities of the outgoing edges of a state must sum up to one, every leaf of $M$ has an implicit outgoing edge connecting it to itself with probability 1. We can now assign probabilities to repairs as follows

$$
\begin{array}{lll}
D - \{\text{Pref}(a, b), \text{Pref}(a, c)\} & \text{has probability} & 2/9 \cdot 1/3 \ + \ 1/9 \cdot 2/4. \\
D - \{\text{Pref}(a, b), \text{Pref}(c, a)\} & \text{has probability} & 2/9 \cdot 2/3 \ + \ 3/9 \cdot 2/5. \\
D - \{\text{Pref}(b, a), \text{Pref}(a, c)\} & \text{has probability} & 3/9 \cdot 1/4 \ + \ 1/9 \cdot 2/4. \\
D - \{\text{Pref}(b, a), \text{Pref}(c, a)\} & \text{has probability} & 3/9 \cdot 3/4 \ + \ 3/9 \cdot 3/5.
\end{array}
$$

For example, the probability of the repair $D - \{\text{Pref}(b, a), \text{Pref}(c, a)\}$ is the probability that the initial state $\varepsilon$ reaches the state $-(b, a), -(c, a)$, i.e., $2/9 \cdot 3/4$ plus the probability that $\varepsilon$ reaches the state $-(c, a), -(b, a)$, i.e., $3/9 \cdot 3/5$.

Let us now explain how flexible query answering based on the likelihood of a tuple appearing in an answer on different repairs is achieved. Consider the query $Q(x) = \forall y \, \text{Pref}(x, y) \vee x = y$ asking for the most preferred products. Observe that in three out of the four repairs shown above such most preferred product does not exist. However, in the last of the repairs, $\{\text{Pref}(a, b), \text{Pref}(a, c), \text{Pref}(a, d), \text{Pref}(b, d)\}$, such a product does exist, namely $a$. Thus, the pair $(a, 0.45)$ is an answer to our query stating that $a$ is a consistent answer with probability $0.45$. In case that a tuple is true in more than one repairs, then we simply sum up the probabilities of the repairs in which the candidate tuple is true. This information on the degree of certainty that $a$ is preferred over all the other products is something that the traditional approach to consistent query answering cannot provide us with. In fact, the set of the certain answers to $Q$ over $D$ under the standard consistent query answering semantics is empty.

## 3 Outline of Main Results

We provide a formalization of the notions informally presented earlier: constraints and their violations, operations, repairing sequences, and Markov chains on such repairing sequences that let us compare their relative importance. This culminates in the definition of an operational repair and a new semantics of query answering based on the degree of certainty that a tuple is in the answer. Since operational repairs have probabilities assigned to them, this degree of certainty is formally defined as a conditional probability that a tuple is in the query answer, under the condition that the database on which the query is asked is an operational repair. We then pinpoint the data complexity of query answering: $\text{FP}^{\#\text{P}}$-complete. We point out that the upper bound relies on the fact that the Markov chain is not explicitly constructed, but is encoded as a function.

With this bound one looks for approximations, and given the probabilistic nature of query answers, we look for approximations via randomized algorithms. There are two

types of guarantees for calculating the conditional probability $p$ of a tuple $\bar{t}$ by means of a randomized algorithm that returns a number $a$. Either $|a-p| \leq \epsilon \cdot p$, for an arbitrary $\epsilon > 0$ (multiplicative error guarantees), or $|a-p| \leq \epsilon$ (additive error guarantees). Since $a$ is the output of a randomized algorithm, we require these to hold with a high probability, say at least $1-\delta$ for small $1 > \delta > 0$. Multiplicative error algorithms (so-called FPRAS: fully polynomial-time randomized approximation scheme) are more common in the literature since the *relative* error between the output of an FPRAS and the value we want to approximate is bounded by $\epsilon$. In the case of additive error algorithms, only the *absolute* error is bounded by $\epsilon$, whereas the relative error increases as the value we want to approximate decreases. Nevertheless, additive error algorithms are equally useful for our purposes since we are approximating probabilities (i.e., the probability of a tuple being in the query answer). Thus, having a high relative error for tuples with small probability is a reasonable price to pay, since such tuples are much less important than tuples with high probability.

We establish two results: our query answering problem does not admit an FPRAS (under some widely believed complexity-theoretic assumptions), but it does admit a polynomial-time randomized approximation algorithm with additive error guarantees. The latter result relies on the fact that we can efficiently sample from the probability space defined by the absorbing states of the given Markov chain. This actually holds when the Markov chain does not admit failing sequences of updates, i.e., sequences that cannot be extended but do not yet repair the database. This is a common occurrence and it covers such common cases as key (or, more generally, EGD) violations.

**Acknowledgements.** We thank the anonymous referees for their useful feedback. This work was supported by the EPSRC Programme Grant EP/M025268/ "VADA: Value Added Data Systems - Principles and Architecture".

## References

1. Marcelo Arenas, Leopoldo E. Bertossi, and Jan Chomicki. Consistent query answers in inconsistent databases. In *PODS*, pages 68–79, 1999.
2. Marco Calautti, Leonid Libkin, and Andreas Pieris. An operational approach to consistent query answering. In *PODS*, 2018. To appear.
3. Jan Chomicki and Jerzy Marcinkowski. Minimal-change integrity maintenance using tuple deletions. *Inf. Comput.*, 197(1-2):90–121, 2005.
4. Ariel Fuxman, Elham Fazli, and Renée J. Miller. Conquer: Efficient management of inconsistent databases. In *SIGMOD*, pages 155–166, 2005.
5. Richard M. Karp and Richard J. Lipton. Some connections between nonuniform and uniform complexity classes. In *STOC*, pages 302–309, 1980.
6. Paraschos Koutris and Dan Suciu. A dichotomy on the complexity of consistent query answering for atoms with simple keys. In *ICDT*, pages 165–176, 2014.
7. Paraschos Koutris and Jef Wijsen. The data complexity of consistent query answering for self-join-free conjunctive queries under primary key constraints. In *PODS*, pages 17–29, 2015.
8. Nicola Leone et al. The INFOMIX system for advanced integration of incomplete and inconsistent data. In *SIGMOD*, pages 915–917, 2005.
9. Jef Wijsen. A survey of the data complexity of consistent query answering under key constraints. In *FoIKS*, pages 62–78, 2014.