Uniform Operational Consistent Query Answering

Marco Calautti
University of Trento
marco.calautti@unitn.it

Andreas Pieris
University of Edinburgh &
University of Cyprus
apieris@inf.ed.ac.uk

Ester Livshits
University of Edinburgh
ester.livshits@ed.ac.uk

Markus Schneider
University of Edinburgh
m.schneider@ed.ac.uk

ABSTRACT
Operational consistent query answering (CQA) is a recent framework for CQA, based on revised definitions of repairs and consistent answers, which opens up the possibility of efficient approximations with explicit error guarantees. The main idea is to iteratively apply operations (e.g., fact deletions), starting from an inconsistent database, until we reach a database that is consistent w.r.t. the given set of constraints. This gives us the flexibility of choosing the probability with which we apply an operation, which in turn allows us to calculate the probability of an operational repair, and thus, the probability with which a consistent answer is entailed. A natural way of assigning probabilities to operations is by targeting the uniform probability distribution over a reasonable space such as the set of operational repairs, the set of sequences of operations that lead to an operational repair, and the set of available operations at a certain step of the repairing process. This leads to what we generally call uniform operational CQA. The goal of this work is to perform a data complexity analysis of both exact and approximate uniform operational CQA, focusing on functional dependencies (and subclasses thereof), and conjunctive queries. The main outcome of our analysis (among other positive and negative results), is that uniform operational CQA pushes the efficiency boundaries further by ensuring the existence of efficient approximation schemes in scenarios that go beyond the simple case of primary keys, which seems to be the limit of the classical approach to CQA.

CCS CONCEPTS
• Information systems → Data management systems.

KEYWORDS
inconsistency; consistent query answering; operational semantics; functional dependencies; conjunctive queries; randomized approximation schemes

1 INTRODUCTION
Consistent query answering (CQA) is an elegant framework, introduced in the late 1990s by Arenas, Bertossi, and Chomicki [1], that allows us to compute conceptually meaningful answers to queries posed over inconsistent databases, that is, databases that do not conform to their specifications. The key elements underlying CQA are (i) the notion of (database) repair of an inconsistent database D, that is, a consistent database whose difference with D is somehow minimal, and (ii) the notion of query answering based on consistent answers, that is, answers that are entailed by every repair. Since deciding whether a candidate answer is a consistent answer is most commonly intractable in data complexity (in fact, even for primary keys and conjunctive queries, the problem is coNP-hard [7]), there was a great effort on drawing the tractability boundary for CQA; see, e.g., [11–13, 16–18]. Much of this effort led to interesting dichotomy results that precisely characterize when CQA is tractable/intractable in data complexity. However, the tractable fragments do not cover many relevant scenarios that go beyond primary keys.

As extensively argued in [5], the goal of a practically applicable CQA approach should be efficient approximate query answering with explicit error guarantees rather than exact query answering. In the realm of the CQA approach described above, one could try to devise efficient probabilistic algorithms with bounded one- or two-sided error. However, it is unlikely that such algorithms exist since, even for very simple scenarios (e.g., primary keys and conjunctive queries), placing the problem in tractable randomized complexity classes such as RP or BPP would imply that the polynomial hierarchy collapses [14]. Another promising idea is to replace the rather strict notion of consistent answers with the more refined notion of relative frequency, that is, the percentage of repairs that entail an answer, and then try to approximate it via a fully polynomial-time randomized approximation scheme (FPRAS); computing it exactly is, unsurprisingly, not P-hard [19]. Indeed, for primary keys and conjunctive queries, one can approximate the relative frequency via an FPRAS; this is implicit in [9], and it has been made explicit in [3]. Moreover, a recent experimental evaluation revealed that approximate CQA in the presence of primary keys and conjunctive queries is not unrealistic in practice [4]. However, it seems that the simple
case of primary keys is the limit of this approach. We have strong indications that in the case of FDs the problem of computing the relative frequency does not admit an FPRAS, while in the case of keys it is a highly non-trivial problem [6].

The above limitations of the classical CQA approach led the authors of [5] to propose a new framework for CQA, based on revised definitions of repairs and consistent answers, which opens up the possibility of efficient approximations with error guarantees. The main idea underlying this new framework is to replace the declarative approach to repairs with an operational one that explains the process of constructing a repair. In other words, we can iteratively apply operations (e.g., fact deletions), starting from an inconsistent database, until we reach a database that is consistent w.r.t. the given set of constraints. This gives us the flexibility of choosing the probability with which we apply an operation, which in turn allows us to calculate the probability of an operational repair, and thus, the probability with which an answer is entailed. Probabilities can be naturally assigned to operations in many scenarios leading to inconsistencies. This is illustrated using an example from [5]:

Example 1.1. Consider a data integration scenario that results in a database containing the facts Emp(1, Alice) and Emp(1, Tom) that violate the constraint that the first attribute of the relation name Emp (the id) is a key. Suppose we have a level of trust in each of the sources; say we believe that each is 50% reliable. With probability 0.5 or 0.5 = 0.25 we do not trust either tuple and apply the operation that removes both facts. With probability (1 − 0.25)/2 = 0.375 we remove either Emp(1, Alice) or Emp(1, Tom). Therefore, we have three repairs, each with its probability: the empty repair with probability 0.25, and the repairs consisting of Emp(1, Alice) or Emp(1, Tom) with probability 0.375. Note that the standard CQA approach only allows the removal of one of the two facts (with equal probability 0.5). It somehow assumes that we trust at least one of the sources, even though they are in conflict.

The preliminary data complexity analysis of operational CQA performed in [5] revealed that computing the probability of a candidate answer is \#P-hard and inapproximable, even for primary keys and conjunctive queries. However, these negative results should not be seen as the end of the story, but rather as the beginning since they are focused on uniform repairs and sequences of operations (i) and (ii) discussed above does not lead (or it remains open whether it leads) to the approximability of operational CQA. The approach of uniform operations renders the problem approximable. The latter is a significant result since it goes beyond the simple case of primary keys.

2 PRELIMINARIES

We recall the basics on relational databases, functional dependencies, and conjunctive queries. In the rest of the paper, we assume the disjoint countably infinite sets \( C \) and \( V \) of constants and variables, respectively. For \( n > 0 \), let \( \{n\} \) be the set \( \{1, \ldots, n\} \).

Relational Databases. A (relational) schema \( S \) is a finite set of relation names with associated arity; we write \( R/n \) to denote that \( R \) has arity \( n \). Each relation name \( R/n \) is associated with a tuple of distinct attribute names \( \{A_1, \ldots, A_n\} \); we write \( \text{att}(R) \) for the set \( \{A_1, \ldots, A_n\} \) of attributes. A fact over \( S \) is an expression of the form \( R(c_1, \ldots, c_n) \), where \( R/n \in S \) and \( c_i \in C \) for each \( i \in \{n\} \). A database \( D \) over \( S \) is a finite set of facts over \( S \). The active domain of \( D \), denoted \( \text{dom}(D) \), is the set of constants occurring in \( D \). In a fact \( f = R(c_1, \ldots, c_n) \), with \( \{A_1, \ldots, A_n\} \) being the tuple of attribute names of \( R \), we write \( f[A_i] \) for the constant \( c_i \).

Functional Dependencies. A functional dependency (FD) \( \phi \) over a schema \( S \) is an expression of the form \( R : X \rightarrow Y \), where \( R/n \in S \) and \( X, Y \subseteq \text{att}(R) \). When \( X \) or \( Y \) are singletons, we avoid the curly braces, and simply write the attribute name. We call \( \phi \) a key if \( X = \text{att}(R) \). Given a set \( \Sigma \) of keys over \( S \), we say that \( \Sigma \) is a set of primary keys if, for every \( R/n \in S \), there exists at most one key in \( \Sigma \) of the form \( R : X \rightarrow Y \). A database \( D \) satisfies an FD \( \phi = R : X \rightarrow Y \), denoted \( D \models \phi \), if, for every two facts \( R(\bar{c}_1), R(\bar{c}_2) \in D \) the following holds: \( R(\bar{c}_1)[A] = R(\bar{c}_2)[A] \) for every \( A \in X \) implies \( R(\bar{c}_1)[B] = R(\bar{c}_2)[B] \) for every \( B \in Y \). We say that \( D \) is consistent w.r.t. a set \( \Sigma \) of FDs, written \( D \models \Sigma \), if \( D \models \phi \) for every \( \phi \in \Sigma \); otherwise, we say that \( D \) is inconsistent w.r.t. \( \Sigma \).

Conjunctive Queries. A (relational) atom \( \alpha \) over a schema \( S \) is an expression of the form \( R(t_1, \ldots, t_n) \), where \( R/n \in S \) and \( t_i \in C \cup V \) for each \( i \in \{n\} \). A conjunctive query (CQ) \( Q \) over \( S \) is an expression of the form \( \text{Ans}(\bar{x}) : R_1(\bar{y}_1) \cdots R_m(\bar{y}_m) \), where \( R_i(\bar{y}_i) \), for \( i \in \{n\} \), is an atom over \( S \), \( \bar{x} \) are the answer variables of \( Q \), and each variable in \( \bar{x} \) is mentioned in \( \bar{y}_i \) for some \( i \in \{n\} \). We may write \( Q(\bar{x}) \) to indicate that \( \bar{x} \) are the answer variables of \( Q \). When \( \bar{x} \) is empty, \( Q \) is called Boolean. The semantics of CQs is given via homomorphisms. Let \( \varphi(Q) \) and \( \text{const}(Q) \) be the set of variables and constants in \( Q \), respectively. A homomorphism from a CQ \( Q \) of the form \( \text{Ans}(\bar{x}) : R_1(\bar{y}_1) \cdots R_n(\bar{y}_n) \) to a database \( D \) is a function...
write we need to keep track of all the reasons that cause the inconsistency by \( V \subseteq S \) schema the set operation to be fixing, i.e., to fix at least one violation, we also need due to the facts \( D \) of \( S \), \( D \) main idea of the operational approach to CQA is to iteratively apply the operations are standard updates +F that add a set \( F \) of facts to the database, or \( -F \) that remove \( F \) from the database. However, since in this work we deal with FDs, we only need to remove facts because the addition of a fact would never resolve a conflict. The formal definition of the notion of operation follows. As usual, we write \( P(S) \) for the powerset of a set \( S \).

Definition 3.1 (Operation). For a database \( D \) over a schema \( S \), a \( D \)-operation is a function \( op : P(D) \rightarrow P(D) \) such that, for some non-empty set \( F \subseteq D \) of facts, for every \( D' \in P(D) \), \( op(D') = D' \setminus F \). We write \( -F \) to refer to this operation.

The operations \( -F \) depend on the database \( D \) as they are defined over \( D \). Since \( D \) will be clear from the context, we may refer to them simply as operations, omitting \( D \). Also, when \( F \) contains a single fact \( f \), we write \( -f \) instead of the more formal \( -\{ f \} \). The main idea of the operational approach to CQA is to iteratively apply operations, starting from an inconsistent database \( D \), until we reach a database \( D' \subseteq D \) that is consistent w.r.t. the given set \( \Sigma \) of FDs. However, as discussed in [5], we need to ensure that at each step of this repairing process, at least one violation is resolved. To this end, we need to keep track of all the reasons that cause the inconsistency of \( D \) \( \Sigma \). This brings us to the notion of FD violation.

Definition 3.2 (FD Violation). For a database \( D \) over a schema \( S \), an \( \Sigma \)-violating of an FD \( \phi \in \Sigma \) over \( S \) is a set \( \{ f, g \} \subseteq D \) of facts such that \( \{ f, g \} \nsubseteq D \). We denote the set of \( \Sigma \)-violations of \( \phi \) by \( V(D, \phi) \). Furthermore, for a set \( \Sigma \) of FDs, we denote by \( V(D, \Sigma) \) the set \( \{ \phi \in \Sigma \mid \phi \in V(D, \phi) \} \).

Thus, a pair \( \phi \in V(D, \Sigma) \) means that one of the reasons why the database \( D \) is inconsistent w.r.t. \( \Sigma \) is because it violates \( \phi \) due to the facts \( f \) and \( g \). As discussed in [5], apart from forcing an operation to be fixing, i.e., to fix at least one violation, we also need to force an operation to remove a set of facts only if it contributes as a whole to a violation. Such operations are called justified.

Definition 3.3 (Justified Operation). Let \( D \) be a database over a schema \( S \), and \( \Sigma \) a set of FDs over \( S \). For a database \( D' \subseteq D \), a \( D \)-operation \( -F \) is called \( (D', \Sigma) \)-justified if there exists \( \phi \in V(D', \Sigma) \) such that \( F \subseteq \{ f, g \} \).

Note that justified operations do not try to minimize the number of atoms that need to be removed. As argued in [5], a set of facts that collectively contributes to a violation should be considered as a justified operation during the iterative repairing process since we do not know a priori which atoms should be deleted, and therefore, we need to explore all the possible scenarios.

### 3 OPERATIONAL CQA

We proceed to recall the recent operational approach to consistent query answering, introduced in [5]. Although this new framework can deal with arbitrary integrity constraints (i.e., tuple-generating dependencies, equality-generating dependencies, and denial constraints), for our purposes we need its simplified version that only deals with functional dependencies.

**Operations and Violations.** The notion of operation is the building block of the operational approach. In the original framework, the operations are standard updates +F that add a set \( F \) of facts to the database, or \( -F \) that remove \( F \) from the database. However, since in this work we deal with FDs, we only need to remove facts because the addition of a fact would never resolve a conflict. The formal definition of the notion of operation follows. As usual, we write \( P(S) \) for the powerset of a set \( S \).

**Definition 3.1 (Operation).** For a database \( D \) over a schema \( S \), a \( D \)-operation is a function \( op : P(D) \rightarrow P(D) \) such that, for some non-empty set \( F \subseteq D \) of facts, for every \( D' \in P(D) \), \( op(D') = D' \setminus F \). We write \( -F \) to refer to this operation.

The operations \( -F \) depend on the database \( D \) as they are defined over \( D \). Since \( D \) will be clear from the context, we may refer to them simply as operations, omitting \( D \). Also, when \( F \) contains a single fact \( f \), we write \( -f \) instead of the more formal \( -\{ f \} \). The main idea of the operational approach to CQA is to iteratively apply operations, starting from an inconsistent database \( D \), until we reach a database \( D' \subseteq D \) that is consistent w.r.t. the given set \( \Sigma \) of FDs. However, as discussed in [5], we need to ensure that at each step of this repairing process, at least one violation is resolved. To this end, we need to keep track of all the reasons that cause the inconsistency of \( D \) w.r.t. \( \Sigma \). This brings us to the notion of FD violation.

**Definition 3.2 (FD Violation).** For a database \( D \) over a schema \( S \), a \( \Sigma \)-violating of an FD \( \phi \in \Sigma \) over \( S \) is a set \( \{ f, g \} \subseteq D \) of facts such that \( \{ f, g \} \nsubseteq D \). We denote the set of \( \Sigma \)-violations of \( \phi \) by \( V(D, \phi) \). Furthermore, for a set \( \Sigma \) of FDs, we denote by \( V(D, \Sigma) \) the set \( \{ \phi \in \Sigma \mid \phi \in V(D, \phi) \} \).

Thus, a pair \( \phi \in V(D, \Sigma) \) means that one of the reasons why the database \( D \) is inconsistent w.r.t. \( \Sigma \) is because it violates \( \phi \) due to the facts \( f \) and \( g \). As discussed in [5], apart from forcing an operation to be fixing, i.e., to fix at least one violation, we also need to force an operation to remove a set of facts only if it contributes as a whole to a violation. Such operations are called justified.

**Definition 3.3 (Justified Operation).** Let \( D \) be a database over a schema \( S \), and \( \Sigma \) a set of FDs over \( S \). For a database \( D' \subseteq D \), a \( D \)-operation \( -F \) is called \( (D', \Sigma) \)-justified if there exists \( \phi \in V(D', \Sigma) \) such that \( F \subseteq \{ f, g \} \).

Note that justified operations do not try to minimize the number of atoms that need to be removed. As argued in [5], a set of facts
Let us clarify that, for a database $D$, and a set $\Sigma$ of FDs, we assume that a $(D, \Sigma)$-repairing Markov chain $(V, E, P)$ is compactly represented as a function $f : RS(D, \Sigma) \times RS(D, \Sigma) \rightarrow [0, 1] \cup \{\bot\}$ such that, for every pair $(s, s') \in RS(D, \Sigma) \times RS(D, \Sigma)$, $f(s, s') = P(s, s')$ if $(s, s') \in E$; otherwise, $f(s, s') = \bot$. We give a simple example, which will serve as a running example, that illustrates the notion of repairing Markov chain:

Example 3.6. Consider the database $D = \{f_1, f_2, f_3\}$ over the schema $S = \{R/3\}$, where $f_1 = R(a_1, b_1, c_1)$, $f_2 = R(a_2, b_2, c_2)$ and $f_3 = R(a_3, b_3, c_3)$. Consider also the set $\Sigma = \{\phi_1, \phi_2\}$ of FDs over $S$, where $\phi_1 = R : A \rightarrow B$ and $\phi_2 = C \rightarrow B$, assuming that $(A, B, C)$ is the tuple of attributes of $R$. It is easy to see that $D \not\models \Sigma$. In particular, we have that $(V, D, \Sigma) = \{((\phi_1, (f_1, f_2)), (\phi_2, (f_2, f_3)))\}$. It is easy to verify that for the edge-labeled rooted tree $T = (V, E, P)$ in Figure 1, $V = RS(D, \Sigma)$, for a non-leaf node $s$ the set of its children is $\text{Ops}_s(D, \Sigma)$, and the set of leaves coincides with $\text{CRS}(D, \Sigma)$. Hence, providing that $p_1 + p_2 + p_3 + p_4 + p_5 = 1$, $p_6 + p_7 + p_8 = 1$ and $p_9 + p_{10} + p_{11} = 1$, $T$ is a $(D, \Sigma)$-repairing Markov chain.

The purpose of a repairing Markov chain generator is to provide a mechanism for defining a family of repairing Markov chains independently of the database. One can design a repairing Markov chain generator $M_\Sigma$ once, and whenever the database $D$ changes, the desired $(D, \Sigma)$-repairing Markov chain is simply $M_\Sigma(D)$.

We now recall the notion of operational repair: they are candidate operational repairs obtained via repairing sequences that are reachable leaves of a repairing Markov chain, i.e., leaves with non-zero probability. The probability of a leaf is coming from the so-called leaf distribution of a repairing Markov chain. Formally, given a database $D$ and a set $\Sigma$ of FDs, the leaf distribution of a $(D, \Sigma)$-repairing Markov chain $T = (V, E, P)$ is a function $p$ that assigns to each leaf $s$ of $T$ a number from $[0, 1]$ as follows: assuming that $(s_0, s_1), (s_1, s_2), \ldots, (s_{n-1}, s_n)$, where $n \geq 0$, $s = s_0$ and $s = s_n$, is the unique path in $T$ from $s$ to $s$, $p(s) = P(s_0, s_1) \cdot P(s_1, s_2) \cdots \cdot P(s_{n-1}, s_n)$. The set of reachable leaves of $T$, denoted $\text{RL}(T)$, is the set of leaves of $T$ that have non-zero probability according to the leaf distribution of $T$.

Definition 3.7 (Operational Repair). Given a database $D$, a set $\Sigma$ of FDs, and a repairing Markov chain generator $M_\Sigma$ w.r.t. $\Sigma$, an (operational) repair of $D$ w.r.t. $M_\Sigma$ is a database $D' \in \text{OCR}(D, \Sigma)$ such that $D' = s(D)$ for some $s \in \text{RL}(M_\Sigma(D))$. Let $\text{OCR}(D, M_\Sigma)$ be the set of all operational repairs of $D$ w.r.t. $M_\Sigma$.

An operational repair may be obtained via multiple repairing sequences that are reachable leaves of the underlying repairing Markov chain. The probability of a repair $D'$ is calculated by summing up the probabilities of all reachable leaves $s$ so that $D' = s(D)$.

Definition 3.8 (Operational Semantics). Given a database $D$, a set $\Sigma$ of FDs, and a repairing Markov chain generator $M_\Sigma$ w.r.t. $\Sigma$, the probability of an operational repair $D'$ of $D$ w.r.t. $M_\Sigma$ is

$$P_{D, M_\Sigma}(D') = \sum_{s \in \text{RL}(M_\Sigma(D)) \text{ and } D' = s(D)} \pi(s),$$

where $\pi$ is the leaf distribution of $M_\Sigma(D)$. The operational semantics of $D$ w.r.t. $M_\Sigma$ is defined as the set of repair-probability pairs $\|D\|_{M_\Sigma} = \{(D', P_{D, M_\Sigma}(D')) | D' \in \text{OCR}(D, M_\Sigma)\}$.

Operational CQA. We now have in place all the necessary notions to recall the operational approach to consistent query answering, and define the main problem of interest. For a database $D$, a set $\Sigma$ of FDs, a Markov chain generator $M_\Sigma$ w.r.t. $\Sigma$, a query $Q(\bar{x})$, and a tuple $\bar{c} \in \text{dom}(D)|^{|\bar{x}|}$, the probability of $\bar{c}$ being an answer to $Q$ over some operational repair of $D$ is defined as

$$P_{M_\Sigma, Q}(D, \bar{c}) = \sum_{(D', P_{D, M_\Sigma}(D')) \in \|D\|_{M_\Sigma}, \bar{c} \in Q(D')} 1/P.$$
In the rest of the section, let $D$ and $\Sigma$ be the database and the set of FDs, respectively, from Example 3.6. Recall that any $(D, \Sigma)$-repairing Markov chain looks as the one depicted in Figure 1 with $p_1 + p_2 + p_3 + p_4 + p_5 = 1$, $p_6 + p_7 + p_8 = 1$ and $p_9 + p_{10} + p_{11} = 1$. Thus, the task of understanding how the Markov chain generators $M^u_\Sigma$ and $M^{uo}_{\Sigma}$ should be defined boils down to understanding how the probabilities $p_1, \ldots, p_{11}$ should be calculated by $M^u_\Sigma$ and $M^{uo}_{\Sigma}$ in order to guarantee the properties discussed above. We start by explaining how the probabilities are calculated by $M^{uo}_{\Sigma}$, which will then help us to explain how the probabilities are calculated by $M^u_\Sigma$. We finally discuss $M^{uo}_{\Sigma}$, which is the simplest one.

**Uniform Sequences.** For a sequence $s \in \text{RS}(D, \Sigma)$, let $\text{CRS}(D, \Sigma)$ be the set of all sequences of CRS$(D, \Sigma)$ that have $s$ as a prefix. Thus, $\text{CRS}(D, \Sigma)$ collects the leaves of the subtree rooted at $s$, with $\text{CRS}(D, \Sigma) = \text{CRS}(D, \Sigma)$ being the set of leaves. Hence, for $M^{uo}_{\Sigma}(D) = (V, E, P)$, to induce the uniform distribution over the leaves, it suffices, for $s, s' \in \text{RS}(D, \Sigma)$ with $s' \in \text{Ops}_s(D, \Sigma)$, to let

$$P(s, s') = \frac{|\text{CRS}_s(D, \Sigma)|}{|\text{CRS}(D, \Sigma)|}.$$ 

Observe that

$$|\text{CRS}_s(D, \Sigma)| = 9$$

$$|\text{CRS}_{-f_1}(D, \Sigma)| = |\text{CRS}_{-f_2}(D, \Sigma)| = |\text{CRS}_{-f_2, f_3}(D, \Sigma)| = 1.$$ 

Hence, $p_1 = p_5 = \frac{3}{5}, p_2 = p_3 = p_4 = \frac{1}{5}$. Similarly, we obtain that $p_6 = p_7 = p_8 = \frac{1}{5},$ and $p_9 + p_{10} + p_{11} = \frac{1}{4}$. Thus, $\text{RL}(M^{uo}_{\Sigma}(D)) = \text{CRS}(D, \Sigma)$, and $\pi(s) = \frac{1}{5}$, for each $s \in \text{RL}(M^{uo}_{\Sigma}(D))$, with $\pi$ being the leaf distribution of $M^{uo}_{\Sigma}(D)$, as needed.

**Uniform Repairs.** Since multiple complete sequences can lead to the same database (e.g., $-f_1, -\{f_2, f_3\}$ and $-f_1, -\{f_1, f_2\}$) we would like to have a mechanism that gives non-zero probability to exactly one such sequence. To this end, for each set of complete sequences that lead to the same consistent database, we identify a representative one. We say that a $(D, \Sigma)$-repairing sequence $s \in \text{CRS}(D, \Sigma)$ is **canonical** if there is no $s' \in \text{CRS}(D, \Sigma)$ such that $s(D) = s'(D)$ and $s' < s$ for some arbitrary ordering $<$. Let $\text{CanCRS}(D, \Sigma)$ be the set of all sequences of CRS$(D, \Sigma)$ that are canonical. Furthermore, for a sequence $s \in \text{RS}(D, \Sigma)$, we write $\text{CanCRS}(D, \Sigma)$ for the set of all sequences $s'$ of $\text{CanCRS}(D, \Sigma)$ that have $s$ as a prefix. Hence, for $s \in \text{RS}(D, \Sigma)$, $\text{CanCRS}(D, \Sigma)$ is the set of canonical leaves of the subtree rooted at $s$, with $\text{CanCRS}(D, \Sigma) = \text{CanCRS}(D, \Sigma)$ being the set of canonical leaves of the tree. We can now follow the same approach discussed above for uniform sequences with the key difference that only canonical sequences are considered. In other words, for $M^{uo}_{\Sigma}(D) = (V, E, P)$ to induce the uniform distribution over the set of operational repairs, it suffices, for nodes $s, s' \in \text{RS}(D, \Sigma)$ with $s' \in \text{Ops}_s(D, \Sigma)$, to let

$$P(s, s') = \frac{|\text{CanCRS}_s(D, \Sigma)|}{|\text{CanCRS}(D, \Sigma)|}.$$ 

Notice that $P(s, s')$ is not defined if the subtree $T_s$ rooted at $s$ has no canonical leaves, i.e., $\text{CanCRS}_s(D, \Sigma) = \emptyset$. In this case, none of the leaves of $T_s$ is reachable with non-zero probability, and thus, $P(s, s')$ can get an arbitrary probability, e.g., $\frac{1}{|\text{Ops}_s(D, \Sigma)|}$.

Let us illustrate the above discussion. Assuming, e.g., that for $s, s' \in \text{RS}(D, \Sigma), s < s'$ iff $s$ comes before $s'$ in a depth-first traversal of the tree, we have that $\text{CanCRS}_s(D, \Sigma)$ consists of the sequences

$$-f_1, -f_2, -f_3, -f_1, -\{f_2, f_3\}, -f_2, -\{f_2, f_3\}.$$ 

Therefore, we get that

$$|\text{CanCRS}_s(D, \Sigma)| = 9$$

$$|\text{CanCRS}_{-f_1}(D, \Sigma)| = |\text{CanCRS}_{-f_2}(D, \Sigma)| = |\text{CanCRS}_{-f_1, f_2}(D, \Sigma)| = 1.$$ 

Notice that, unlike the Markov chain generators $M^u_\Sigma$ and $M^{uo}_{\Sigma}$ discussed above, $M^{uo}_{\Sigma}$ is intrinsically “local” in the sense that the probabilities assigned to operations at a certain step are completely determined by that step. As we shall see, the local nature of $M^{uo}_{\Sigma}$ has a significant impact on operational CQA when it comes to approximations.

At this point, we would like to stress that there is an interesting connection between $M^{uo}_{\Sigma}$ and the notion of repair-key, a construct introduced in [15] towards a query algebra for probabilistic databases. It turns out that one can achieve the same as $M^{uo}_{\Sigma}$ by exploiting the repair-key construct. However, this is not useful for our main objective, which, as explained below, is to analyze the complexity of uniform operational CQA. In particular, we cannot inherit any (in)approximability results from the repair-key literature.

**Our Main Objective.** The data complexity of OCQA for arbitrary Markov chain generators has been already studied in [5], showing that it is, in general, intractable. In particular:

**Theorem 4.1 ([5]).** There exist a set $\Sigma$ of primary keys, a repairing Markov chain generator $M_\Sigma$ w.r.t. $\Sigma$, and a CQ $\bar{x}$ such that $\text{OCQA}(\Sigma, M_\Sigma, Q(\bar{x}))$ is $\#P$-hard.

Note that the above result in [5] is stated for arbitrary keys, but a simple inspection of the underlying proof reveals that it holds even for primary keys. With the above intractability result in place, the authors of [5] asked whether $\text{OCQA}(\Sigma, M_\Sigma, Q(\bar{x}))$ is approximable, i.e., whether the target probability can be approximated via a *fully polynomial-time randomized approximation scheme* (FPRAS, for short). An FPRAS for $\text{OCQA}(\Sigma, M_\Sigma, Q(\bar{x}))$ is a randomized algorithm $A$ that takes as input a database $D$, a tuple $\bar{c} \in \text{dom}(D)[\bar{x}]$, $\epsilon > 0$, and $0 < \delta < 1$, runs in polynomial time in $|D|$, $|\bar{c}|$, $1/\epsilon$ and $\log(1/\delta)$, and produces a random variable $A(D, \bar{c}, \epsilon, \delta)$ such that

$$\Pr (|A(D, \bar{c}, \epsilon, \delta) - P_{M_\Sigma, Q}(D, \bar{c})| \leq \epsilon \cdot P_{M_\Sigma, Q}(D, \bar{c})) \geq 1 - \delta.$$ 

It was shown that the problem in question does not admit an FPRAS, under the widely accepted complexity assumption that $\text{RP} \neq \text{NP}$. Recall that $\text{RP}$ is the complexity class of problems that are efficiently

\footnote{As usual, $|\bar{a}|$ denotes the size of the encoding of a syntactic object $a$.}
solvable via a randomized algorithm with a bounded one-sided error (i.e., the answer may mistakenly be "no") [2].

**THEOREM 4.2 ([5]).** Unless \( \text{RP} = \text{NP} \), there exist a set \( \Sigma \) of primary keys, a Markov chain generator \( M^u_\Sigma \) w.r.t. \( \Sigma \), and a CQ \( Q \) such that there is no FPRAS for OCQA(\( \Sigma, M^u_\Sigma, Q \)).

Again, the above result in [5] is stated for arbitrary keys, but it was actually shown for primary keys. Having the natural Markov chain generators discussed above in place, the question is how the complexity of exact and approximate operational CQA is affected, i.e., how Theorems 4.1 and 4.2 are affected if we consider these more refined Markov chain generators instead of an arbitrary one. The goal of this work is to perform such a complexity analysis. Our main findings can be summarized as follows:

1. The complexity of exact operational CQA remains \#P-hard, even in the case of primary keys.
2. Operational CQA is approximable, i.e., it admits an FPRAS, if we focus on primary keys.
3. In the case of arbitrary keys and FDs, although the Markov chain generators based on uniform repairs and sequences do not lead (or it remains open whether they lead) to the approximability of operational CQA, the Markov chain generator based on uniform operations renders the problem approximable. The latter should be attributed to the "local" nature of the Markov chain generator based on uniform operations.

The rest of the paper is devoted to discussing the high-level ideas underlying the above results. But let us first briefly discuss the conceptual relevance of the Markov chain generators in question.

**Justifying Uniform Operational CQA.** It should be clear that the Markov chain generator \( M^u_\Sigma \), based on uniform repairs, is a self-justified choice. Indeed, like in every probabilistic framework, if we do not know how to assign probabilities to the elements of the underlying space (here, candidate operational repairs), adopting the uniform distribution is the obvious way to go.

After showing that with \( M^u_\Sigma \) the problem of interest is inapproximable, the next natural step is to investigate how to assign probabilities to operational repairs in a more refined way by taking into account the repairing process that leads to the operational repairs, which in turn will give more structure to the assigned probabilities that could be exploited towards efficient approximation schemes. In other words, we would like to assign probabilities to operational repairs by taking into account the support in terms of the repairing process. This is completely neglected by \( M^u_\Sigma \) since an operational repair obtained by very few repairing sequences is considered equally important as an operational repair obtained by several repairing sequences. This is precisely what the Markov chain generator \( M_\Sigma \) based on uniform sequences, achieves, that is, it assigns probabilities to operational repairs by taking into account the support in terms of the repairing process.

In the process of getting approximability results beyond the case of primary keys, the next step is to take into account structural properties of the repairing sequences, i.e., to consider some repairing sequences more important than others depending on their structure. Such a structural property is the length of a sequence.

---

*In the case of FDs, the approximability result holds assuming that only operations that remove a single fact (not a pair of facts) are considered, this is discussed in Section 7.*

which in turn translates into the number of modifications applied to the given database. Since, in general, we would like to stay as close as possible to the original database, it is natural to classify shorter sequences as more important than longer ones. This is what the Markov chain generator \( M^u_\Sigma \), based on uniform operations, is aiming at. In other words, \( M^u_\Sigma \) assigns probabilities to repairs by consolidating two parameters: the number of sequences that lead to a repair, and the number of modifications applied to the given database; this is reflected in Example 3.6.

## 5 UNIFORM REPAIRS

We start our complexity analysis by considering the Markov chain generator based on uniform repairs, and show the following result:

**THEOREM 5.1.**

1. There exist a set \( \Sigma \) of primary keys, and a CQ \( Q \) such that OCQA(\( \Sigma, M^u_\Sigma, Q \)) is \#P-hard.
2. For a set \( \Sigma \) of primary keys, and a CQ \( Q \), OCQA(\( \Sigma, M^u_\Sigma, Q \)) admits an FPRAS.
3. Unless \( \text{RP} = \text{NP} \), there exist a set \( \Sigma \) of FDs, and a CQ \( Q \) such that there is no FPRAS for OCQA(\( \Sigma, M^u_\Sigma, Q \)).

Notice that the above result does not cover the case of arbitrary keys, which remains an open problem. We can extract, however, from the proof of item (3) that for keys, unless \( \text{RP} = \text{NP} \), the problem of counting the number of operational repairs does not admit an FPRAS. We see this as an indication that item (3) holds even in the case of keys. We now discuss how Theorem 5.1 is shown.

We start with the simple observation that, for a database \( D \), a set \( \Sigma \) of FDs, a CQ \( \exists ( \bar{x} ) \), and a tuple \( \bar{c} \in \text{dom}(D)[\bar{x}] \),

\[
P_{M^u_\Sigma, Q}(D, \bar{c}) = \frac{|\{d' \in \text{CORep}(D, \Sigma) \mid \bar{c} \in Q(D')\}|}{|\text{CORep}(D, \Sigma)|}.
\]

This ratio is the percentage of candidate operational repairs of \( D \) w.r.t. \( \Sigma \) that entail \( Q(\bar{c}) \), which we call the repair relative frequency of \( Q(\bar{c}) \) w.r.t. \( D \) and \( \Sigma \), and denote \( \text{RRFreq}_{Q}(D, \bar{c}) \). Therefore, we can conveniently restate the problem OCQA(\( \Sigma, M^u_\Sigma, Q \)) as the problem of computing the repair relative frequency of \( Q(\bar{c}) \) w.r.t. \( D \) and \( \Sigma \), which does not depend on the Markov chain generator \( M^u_\Sigma \):

**PROBLEM:** \( \text{RRFreq}(\Sigma, Q(\bar{c})) \)

**INPUT:** A database \( D \), and a tuple \( \bar{c} \in \text{dom}(D)[\bar{x}] \).

**OUTPUT:** \( \text{RRFreq}_{Q}(D, \bar{c}) \).

We proceed to discuss how we establish Theorem 5.1 by directly considering the problem \( \text{RRFreq}(\Sigma, Q(\bar{c})) \) instead of \( \text{OCQA}(\Sigma, M^u_\Sigma, Q) \).

**Item (1).** We show that \( \text{RRFreq}(\Sigma, Q(\bar{c})) \) is \#P-hard for a set \( \Sigma \) consisting of a single key of the form \( R : A \rightarrow B \), where \( R \) is a binary relation name with \( (A, B) \) being its tuple of attributes, and a Boolean CQ \( Q \). This is done via a polynomial-time Turing reduction from a graph-theoretic problem called \( \text{HH}-\text{Coloring} \), where \( H \) is an undirected graph, to \( \text{RRFreq}(\Sigma, Q) \). The problem \( \text{HH}-\text{Coloring} \) takes as input an undirected graph \( G \), and asks for the number of homomorphisms from \( G \) to \( H \). The key of the proof is to carefully choose \( H \) so that (i) \( \text{HH}-\text{Coloring} \) is \#P-hard, and (ii) it allows us to devise the desired polynomial-time Turing reduction, i.e., for an undirected graph \( G \), we can construct in polynomial time in \(|G|\) a database \( D_G \) such that the number of homomorphisms from \( G \) to \( H \) can be computed in polynomial time in \(|G|\), assuming that we have
access to an oracle for the problem $\text{RRFreq}(\Sigma, Q)$, which we can call to compute the number $\text{rrfreq}_{\Sigma, Q}(D_G, \cdot)$; we use $(\cdot)$ to denote the empty tuple. For choosing $H$, we exploit an interesting dichotomy from [10], which characterizes when $\mathcal{H}$-Coloring is solvable in polynomial time or is $\mathbb{P}$-hard, depending on the structure of $H$.

**Item (2).** For showing that, for a set $\Sigma$ of primary keys and a CQ $Q$, $\text{RRFreq}(\Sigma, Q)$ admits an FPRAS, we rely on Monte Carlo sampling. We first show the existence of an efficient sampler:

**Lemma 5.2.** Given a database $D$, and a set $\Sigma$ of primary keys, we can sample elements of $\text{CORep}(D, \Sigma)$ uniformly at random in polynomial time in $|D|$. The above lemma tells us that there exists a randomized algorithm $\text{SampleRep}$ that takes as input $D$ and $\Sigma$, runs in polynomial time in $|D|$, and produces a random variable $\text{SampleRep}(D, \Sigma)$ such that $\Pr(\text{SampleRep}(D, \Sigma) = D' \in \text{CORep}(D, \Sigma))$ for every database $D' \in \text{CORep}(D, \Sigma)$. Notice, however, that the efficient sampler provided by Lemma 5.2 does not immediately imply the existence of an FPRAS for $\text{RRFreq}(\Sigma, Q)$ since the number of samples should be proportional to $\frac{1}{\text{rrfreq}_{\Sigma, Q}(D, \cdot)}$ [8]. Hence, to obtain an FPRAS using Monte Carlo sampling, we need show that the repair relative frequency is never “too small”.

**Lemma 5.3.** Consider a set $\Sigma$ of primary keys, and a CQ $Q(\cdot)$. For every database $D$, and tuple $\bar{c} \in \text{dom}(D)^{|X|}$, 

\[
\text{rrfreq}_{\Sigma, Q}(D, \bar{c}) \geq \frac{1}{(2 \cdot |D|)!|Q|!}
\]

whenever $\text{rrfreq}_{\Sigma, Q}(D, \bar{c}) > 0$.

Given a set $\Sigma$ of primary keys and a CQ $Q$, by exploiting Lemmas 5.2 and 5.3, we can easily devise an FPRAS for $\text{RRFreq}(\Sigma, Q)$.

**Item (3).** For showing that there exist a set $\Sigma$ of FDs and a CQ $Q$ such that, unless $\mathbb{RP} = \mathbb{NP}$, there is no FPRAS for $\text{RRFreq}(\Sigma, Q)$, we provide a rather involved proof that proceeds in two main steps. We first give an auxiliary lemma that is needed by both steps.

An undirected graph $G$ is called non-trivially connected if it contains at least two nodes, and is connected. We write $\text{IS}(G)$ for the set that collects all the independent sets of $G$. Recall that the conflict graph of a database $D$ w.r.t. a set $\Sigma$ of FDs, denoted $CG(D, \Sigma)$, is an undirected graph whose node set is $D$, and it has an edge between $f$ and $g$ if $\{f, g\} \not\in \Sigma$. A database $D$ is non-trivially $\Sigma$-connected if $CG(D, \Sigma)$ is non-trivially connected. We then show the following:

**Lemma 5.4.** Consider a non-trivially $\Sigma$-connected database $D$, where $\Sigma$ is a set of FDs. It holds that $|\text{CORep}(D, \Sigma)| = |	ext{IS}(CG(D, \Sigma))|$. Having the above auxiliary lemma in place, we can now describe the two steps of the proof underlying Theorem 5.1(3). The first step establishes the following inapproximability result about keys.

**Proposition 5.5.** Unless $\mathbb{RP} = \mathbb{NP}$, there exists a set $\Sigma$ of keys over $(R)$ such that, given a non-trivially $\Sigma$-connected database $D$, the problem of computing $|\text{CORep}(D, \Sigma)|$ does not admit an FPRAS.

The above result exploits the fact that, unless $\mathbb{RP} = \mathbb{NP}$, the problem of counting the number of independent sets of a non-trivially connected undirected graph of bounded degree does not admit an FPRAS.$^3$ In particular, we show that there exists a set $\Sigma_K$ of keys over the schema $\Sigma = (R/\Delta + 1)$ such that the following holds: given a non-trivially connected undirected graph $G$ of bounded degree $\Delta$, we can construct in polynomial time in $|G|$ a database $D_G$ over $S$ such that $CG(D_G, \Sigma_K)$ is isomorphic to $G$. Thus, by Lemma 5.4, $|\text{CORep}(D_G, \Sigma_K)| = |\text{IS}(G)|$. The construction of $D_G$ exploits Vizing’s Theorem, which states that a graph of degree $\Delta$ always has a $(\Delta + 1)$-edge-coloring, as well as the fact that such an edge-coloring can be constructed in polynomial time as long as $\Delta$ is bounded [20]. Hence, given a database $D$, assuming that the problem of computing the number $|\text{CORep}(D, \Sigma_K)|$ admits an FPRAS, we can conclude that the problem of counting the number of independent sets of a non-trivially connected undirected graph of bounded degree admits an FPRAS, which, unless $\mathbb{RP} = \mathbb{NP}$, leads to a contradiction. Therefore, Proposition 5.5 follows with $\Sigma = \Sigma_K$. Notice that Proposition 5.5 tells us that, for keys, unless $\mathbb{RP} = \mathbb{NP}$, the problem of counting the number of operational repairs does not admit an FPRAS. As said above, we see this as an indication that item (3) of Theorem 5.1 holds even for keys.

We then proceed to show that, unless $\mathbb{RP} = \mathbb{NP}$, the existence of an FPRAS for $\text{RRFreq}(\Sigma, Q)$, where $\Sigma$ is a set of FDs and $Q$ a CQ, would contradict Proposition 5.5. Let $\Sigma_K$ be the set of keys provided by Proposition 5.5. We show the following auxiliary result:

**Lemma 5.6.** Assume that $\text{RRFreq}(\Sigma, Q)$ admits an FPRAS, for every set $\Sigma$ of FDs and CQ $Q$. Given a non-trivially $\Sigma_K$-connected database $D$, the problem of computing $|\text{CORep}(D, \Sigma_K)|$ admits an FPRAS.

To establish the above result, we show that there exists a set $\Sigma_F$ of FDs such that, for every non-trivially $\Sigma_K$-connected database $D$, we can construct in polynomial time in $|D|$ a database $D_F$ such that $CG(D_F, \Sigma_F)$ consists of a graph $G$ that is isomorphic to $CG(D, \Sigma_K)$, and an additional node that is connected via an edge with every node of $G$. Therefore, by Lemma 5.4, we get that $|\text{CORep}(D_F, \Sigma_F)| = |\text{CORep}(D, \Sigma_K)| + 1$.

Let us clarify that this is the place where we need the power of FDs; it is unclear how we can devise a set of keys that has the same properties as $\Sigma_F$. We then construct an atomic Boolean CQ $Q_F$ with

\[
\text{rrfreq}_{\Sigma_F, Q_F}(D_F, \cdot) = \frac{1}{|\text{CORep}(D_F, \Sigma_F)|} = \frac{1}{|\text{CORep}(D, \Sigma_K)| + 1}.
\]

we use $(\cdot)$ to denote the empty tuple. Now, by exploiting the above equality, the fact that $D_F$ can be constructed in polynomial time, and the FPRAS for $\text{RRFreq}(\Sigma_F, Q_F)$ (which exists by hypothesis), we can devise an FPRAS for the problem of computing $|\text{CORep}(D, \Sigma_K)|$ given a non-trivially $\Sigma_K$-connected database $D$, as claimed.

It is now straightforward to see that from Proposition 5.5 and Lemma 5.6, we get that, unless $\mathbb{RP} = \mathbb{NP}$, there exist a set $\Sigma$ of FDs and a CQ $Q$ such that there is no FPRAS for $\text{RRFreq}(\Sigma, Q)$.

### 6 Uniform Sequences

We now concentrate on the Markov chain generator based on uniform sequences, and establish the following complexity result.

**Theorem 6.1.** (1) There exist a set $\Sigma$ of primary keys, and a CQ $Q$ such that $\text{OCQA}(\Sigma, M_{\Sigma, Q}^\omega) \in \mathbb{P}$-hard.

This result is known for arbitrary, not necessarily non-trivially connected graphs [21]. Thus, for our purposes, we had to strengthen it to non-trivially connected graphs.
(2) For a set $\Sigma$ of primary keys, and a CQ $\mathcal{Q}$, OCQA$(\Sigma, M^{\text{opus}}_{\Sigma}, \mathcal{Q})$ admits an FPRAS.

Notice that the above result does not cover the cases of arbitrary keys and FDs. Unfortunately, despite our efforts, we have not managed to prove or disprove the existence of an FPRAS for the problem in question. We conjecture that there is no FPRAS even for keys, i.e., unless $\text{RP} = \text{NP}$, there exist a set $\Sigma$ of keys, and a CQ $\mathcal{Q}$ such that there is no FPRAS for OCQA$(\Sigma, M^{\text{opus}}_{\Sigma}, \mathcal{Q})$. We proceed to discuss how Theorem 6.1 is shown.

As for Theorem 5.1, we can conveniently restate the problem in question as a problem of computing a "relative frequency" ratio that does not depend on the Markov chain generator. In particular, for a database $D$, a set $\Sigma$ of FDs, a CQ $\mathcal{Q}(\bar{x})$, and a tuple $\bar{c} \in \text{dom}(D)^{|\bar{x}|}$,

$$P_{M^{\text{opus}}_{\Sigma},\mathcal{Q}}(D, \bar{c}) = \frac{|\{ \bar{x} \in \text{CRS}(D, \Sigma) \mid \bar{c} \in \mathcal{Q}(s(D)) \}|}{|\text{CRS}(D, \Sigma)|}.$$  

This ratio is the percentage of complete $(D, \Sigma)$-repairing sequences that lead to an operational repair that entails $\mathcal{Q}(\bar{c})$, which we call the sequence relative frequency of $\mathcal{Q}(\bar{c})$ w.r.t. $D$ and $\Sigma$, and denote $\text{srFreq}_{\Sigma,\mathcal{Q}}(D, \bar{c})$. Thus, we can restate OCQA$(\Sigma, M^{\text{opus}}_{\Sigma}, \mathcal{Q})$ as the problem of computing the sequence relative frequency of $\mathcal{Q}(\bar{c})$ w.r.t. $D$ and $\Sigma$, which is independent from the Markov chain generator $M^{\text{opus}}_{\Sigma}$.

We now discuss how we establish Theorem 6.1 by directly considering the problem SRFreq$(\Sigma, \mathcal{Q})$ instead of OCQA$(\Sigma, M^{\text{opus}}_{\Sigma}, \mathcal{Q})$.

**Item (1).** Let $\Sigma$ and $\mathcal{Q}$ be the singleton set of primary keys and the Boolean CQ, respectively, for which $\text{SRFreq}(\Sigma, \mathcal{Q})$ is $\#\text{P}$-hard; $\Sigma$ and $\mathcal{Q}$ are obtained from the proof of Theorem 5.1(1). We show that also SRFreq$(\Sigma, \mathcal{Q})$ is $\#\text{P}$-hard via a polynomial-time Turing reduction from $\#\text{H}-\text{Coloring}$. Actually, we can exploit the same construction as in the proof of item (1) of Theorem 5.1.

**Item (2).** For showing that, for a set $\Sigma$ of primary keys and a CQ $\mathcal{Q}(\bar{x})$, SRFreq$(\Sigma, \mathcal{Q})$ admits an FPRAS, we rely again on Monte Carlo sampling. We first show that an efficient sampler exists. This relies on a non-trivial technical lemma, which states that, for a database $D$, $|\text{CRS}(D, \Sigma)|$ can be computed in polynomial time in $|D|^{\text{poly}}$.

**Lemma 6.2.** For a database $D$, and a set $\Sigma$ of primary keys, we can sample elements of $\text{CRS}(D, \Sigma)$ uniformly at random in polynomial time in $|D|^{\text{poly}}$.

To establish that the problem in question admits an FPRAS based on Monte Carlo sampling, it remains to show the following:

**Lemma 6.3.** Consider a set $\Sigma$ of primary keys, and a CQ $\mathcal{Q}(\bar{x})$. For every database $D$, and tuple $\bar{c} \in \text{dom}(D)^{|\bar{x}|}$,

$$\text{srFreq}_{\Sigma,\mathcal{Q}}(D, \bar{c}) \geq \frac{1}{(2 \cdot |D|)^{|\bar{c}|}}.$$  

whenever $\text{srFreq}_{\Sigma,\mathcal{Q}}(D, \bar{c}) > 0$.

Given a set $\Sigma$ of primary keys and a CQ $\mathcal{Q}$, by exploiting Lemmas 6.2 and 6.3, we can easily devise an FPRAS for SRFreq$(\Sigma, \mathcal{Q})$.

### 7 Uniform Operations

We finally consider the Markov chain generator based on uniform operations, and establish the following complexity result.

**Theorem 7.1.**

1. There exist a set $\Sigma$ of primary keys, and a CQ $\mathcal{Q}$ such that OCQA$(\Sigma, M^{\text{opus}}_{\Sigma}, \mathcal{Q})$ is $\#\text{P}$-hard.

2. For a set $\Sigma$ of keys, and a CQ $\mathcal{Q}$, OCQA$(\Sigma, M^{\text{opus}}_{\Sigma}, \mathcal{Q})$ admits an FPRAS.

Notice that the above result does not cover the case of FDs, which remains an open problem. However, as we explain below, for FDs we can establish an approximability result under the assumption that only operations that remove a single fact (not a pair of facts) are considered. But let us first discuss the proof of Theorem 7.1.

Unlike Theorems 5.1 and 6.1 presented above, there is no obvious way to conveniently restate the problem of interest as a problem of computing a "relative frequency" ratio. Thus, the proof of Theorem 7.1, which we discuss next, has to deal with OCQA$(\Sigma, M^{\text{opus}}_{\Sigma}, \mathcal{Q})$ for a set $\Sigma$ of FDs and a CQ $\mathcal{Q}$.

**Item (1).** As we did for item (1) of Theorem 6.1, we reuse the construction underlying the proof of item (1) of Theorem 5.1.

**Item (2).** We show that OCQA$(\Sigma, M^{\text{opus}}_{\Sigma}, \mathcal{Q})$, where $\Sigma$ is a set of keys and $\mathcal{Q}$ a CQ, admits an FPRAS by relying once again on Monte Carlo sampling. The existence of an efficient sampler follows easily from the definition of the Markov chain generator $M^{\text{opus}}_{\Sigma}$. In particular:

**Lemma 7.2.** Given a database $D$, and a set $\Sigma$ of keys, we can sample elements of $\text{RL}(M^{\text{opus}}_{\Sigma}(D))$ according to the leaf distribution of $M^{\text{opus}}_{\Sigma}(D)$ in polynomial time in $|D|$.

The interesting task towards an FPRAS for the problem in question is to show that the target probability is never "too small".

**Proposition 7.3.** Consider a set $\Sigma$ of keys, and a CQ $\mathcal{Q}(\bar{x})$. There is a polynomial $\text{pol}$ such that, for every database $D$, and $\bar{c} \in \text{dom}(D)^{|\bar{x}|},$

$$P_{M^{\text{opus}}_{\Sigma},\mathcal{Q}}(D, \bar{c}) \geq \frac{1}{\text{pol}(|D|)}$$  

whenever $P_{M^{\text{opus}}_{\Sigma},\mathcal{Q}}(D, \bar{c}) > 0$.

We proceed to discuss the main ideas underlying the proof of the above result. For the sake of clarity, we focus on atomic queries, i.e., CQs with only one atom. In the sequel, let $\Sigma$ be a set of keys, $\mathcal{Q}(\bar{x})$ an atomic query, $D$ a database, and $\bar{c}$ a tuple of $\text{dom}(D)^{|\bar{x}|}$.

Clearly, if there is no homomorphism $h$ from $Q$ to $D$ with $h(\bar{x}) = \bar{c}$, then $P_{M^{\text{opus}}_{\Sigma},\mathcal{Q}}(D, \bar{c}) = 0$. Assume now that such a homomorphism $h$ exists, and let $f$ be the fact of $D$ obtained after applying $h$ to the single atom of $Q$. It is not difficult to see that

$$P_{M^{\text{opus}}_{\Sigma},\mathcal{Q}}(D, \bar{c}) \geq \sum_{D' \in \text{ORep}(D, M^{\text{opus}}_{\Sigma}) \text{ and } f \in D'} P_{D, M^{\text{opus}}_{\Sigma}}(D').$$

Thus, it suffices to show that there exists a polynomial $\text{pol}$ such that $\Lambda \geq \frac{1}{\text{pol}(|D|)}$. Let $S_{f}$ and $S_{\neg f}$ be the sets of sequences of $\text{RL}(M^{\text{opus}}_{\Sigma}(D))$ that keep $f$ and remove $f$, respectively, i.e.,

$$S_{f} = \{ s \in \text{RL}(M^{\text{opus}}_{\Sigma}(D)) \mid f \in s(D) \}$$

$$S_{\neg f} = \{ s \in \text{RL}(M^{\text{opus}}_{\Sigma}(D)) \mid f \notin s(D) \}.$$
With \( \pi \) being the leaf distribution of \( M_\Sigma^\omega(D) \), \( \Lambda = \frac{\Lambda_f}{\Lambda_f + \Lambda_{\neg f}} \), where 
\[
\Lambda_f = \sum_{s \in S_f} \pi(s) \quad \text{and} \quad \Lambda_{\neg f} = \sum_{s \in S_{\neg f}} \pi(s).
\]

Therefore, to establish the desired lower bound \( \frac{1}{\pol(||D||)} \) for \( \Lambda \), it suffices to show that there exists a polynomial \( \pol' \) such that \( \Lambda_{\neg f} \leq \pol'(||D||) \cdot \Lambda_f \). Indeed, in this case we can conclude that 
\[
\Lambda = \frac{\Lambda_f}{\Lambda_f + \Lambda_{\neg f}} \geq \frac{\Lambda_f}{\Lambda_f + \pol'(||D||) \cdot \Lambda_f} = \frac{1}{1 + \pol'(||D||)},
\]
and the claim follows with \( \pol'(||D||) = \pol''(||D||) \cdot (2 \cdot ||D|| - 1) \).

**An FPRAS for FDs.** Recall that Theorem 7.1 does not cover the case of FDs, which remains an open problem. At this point, one may wonder whether Monte Carlo sampling can be used for devising an FPRAS in the case of FDs. Indeed, the efficient sampler provided by Lemma 7.2 holds even for FDs since the proof of that lemma does not exploit keys in any way, but only the “local” nature of the Markov chain generator. However, we do not have a result analogous to Proposition 7.3, which states that the target probability is never “too small”. In fact, there exist a set \( \Sigma \) of FDs, a Boolean atomic query \( Q \), and a family of databases \( \{D_n\}_{n \geq 0} \) with \( |D_n| = n \), such that \( 0 < \Pr_{M_{\Sigma}^\omega,Q}(D_n,()) \leq \frac{1}{n^2} \). Hence, for devising an FPRAS in the case of FDs (if it exists), we need a more sophisticated machinery than the one based on Monte Carlo sampling. On the other hand, we can establish a result analogous to Proposition 7.3 for FDs, assuming that only operations that remove a single fact (not a pair of facts) are considered. Given a set \( \Sigma \) of FDs, let \( M_{\Sigma}^\omega,1 \) be the Markov chain generator defined as \( M_{\Sigma}^\omega \), with the difference that only sequences consisting of operations that remove a single fact are considered. We then get the following based on Monte Carlo sampling:

**Theorem 7.5.** For any set \( \Sigma \) of FDs, and a CQA \( \text{OCQA}(\Sigma, M_{\Sigma}^\omega,1, Q) \) admits an FPRAS.

Note that singleton operations do not alter the data complexity of exact operational CQA; we can show that item (1) of Theorem 7.1 continues to hold. Let us also clarify that focusing on singleton operations does not affect Theorem 5.1 and Theorem 6.1.

**8 CONCLUSIONS**

The take-home message of our work is that uniform operational CQA is flexible enough to lead to approximability results that go beyond the simple case of primary keys, which seems to be the limit of the classical approach to CQA.

Although we understand well uniform operational CQA, there are still interesting open problems concerning approximability:

1. the case of keys and uniform repairs (we only have a negative result for the problem of counting repairs),
2. the case of keys/FDs and uniform sequences, and
3. the case of FDs and uniform operations (we only have a positive result assuming singleton operations).

Another interesting direction for future research is to consider conceptually relevant distributions that deviate from the uniform ones considered in this work, and perform the same complexity analysis of exact and approximate operational CQA.

**Acknowledgements.** We thank the referees for their feedback. This work was supported by the EPSRC grant EP/S003800/1 "EQUID", by the EPSRC Centre for Doctoral Training in Data Science (grant EP/L016427/1), and the University of Edinburgh.

**REFERENCES**

[1] Marcelo Arenas, Leopoldo E. Bertossi, and Jan Chomicki. 1999. Consistent Query Answers in Inconsistent Databases. In PODS 68–79.
[2] Sanjeev Arora and Boaz Barak. 2009. Computational Complexity: A Modern Approach. Cambridge University Press.
[3] Marco Calautti, Marco Console, and Andreas Pieris. 2019. Counting Database Updates under Primary Keys Revisited. In PODS 104–118.
[4] Marco Calautti, Marco Console, and Andreas Pieris. 2021. Benchmarking Approximate Consistent Query Answering. In PODS 233–246.

[5] Marco Calautti, Leonid Libkin, and Andreas Pieris. 2018. An Operational Approach to Consistent Query Answering. In PODS 239–251.

[6] Marco Calautti, Ester Livshits, Andreas Pieris, and Markus Schneider. 2022. Counting Database Repairs Entailing a Query: The Case of Functional Dependencies. In PODS. To appear.

[7] Jan Chomicki and Jerzy Marcinkowski. 2005. Minimal-change integrity maintenance using tuple deletions. Inf. Comput. 197, 1-2 (2005), 90–121.

[8] Paul Dagum, Richard M. Karp, Michael Luby, and Sheldon M. Ross. 2000. An Optimal Algorithm for Monte Carlo Estimation. SIAM J. Comput. 29, 5 (2000), 1484–1496.

[9] Nilesh N. Dalvi and Dan Suciu. 2007. Management of probabilistic data: foundations and challenges. In PODS. 1–12.

[10] Martin E. Dyer and Catherine S. Greenhill. 2000. The complexity of counting graph homomorphisms. Random Struct. Algorithms 17, 3-4 (2000), 260–289.

[11] Ariel Fuxman, Elham Fazli, and Renée J. Miller. 2005. ConQuer: Efficient Management of Inconsistent Databases. In SIGMOD. 155–166.

[12] Ariel Fuxman and Renée J. Miller. 2007. First-order query rewriting for inconsistent databases. J. Comput. Syst. Sci. 73, 4 (2007), 610–635.

[13] Floris Geerts, Fabian Pijck, and Jef Wijsen. 2015. First-Order Under-Approximations of Consistent Query Answers. In SIM. 354–367.

[14] Richard M. Karp and Richard J. Lipton. 1980. Some Connections between Nonuniform and Uniform Complexity Classes. In STOC. 302–309.

[15] Christoph Koch. 2008. On Query Algebras for Probabilistic Databases. SIGMOD Rec. 37, 4 (2008), 78–85.

[16] Paraschos Koutris and Dan Suciu. 2014. A Dichotomy on the Complexity of Consistent Query Answering for Atoms with Simple Keys. In ICDT. 165–176.

[17] Paraschos Koutris and Jef Wijsen. 2015. The Data Complexity of Consistent Query Answering for Self-Join-Free Conjunctive Queries Under Primary Key Constraints. In PODS. 17–29.

[18] Paraschos Koutris and Jef Wijsen. 2021. Consistent Query Answering for Primary Keys in Datalog. Theory Comput. Syst. 65, 1 (2021), 122–178.

[19] Dany Maslowski and Jef Wijsen. 2013. A dichotomy in the complexity of counting database repairs. J. Comput. Syst. Sci. 79, 6 (2013), 956–983.

[20] Jayadev Micr and David Gries. 1992. A constructive proof of Vizing’s theorem. Inform. Process. Lett. 41, 3 (1992), 131–133.

[21] Allan Sly. 2010. Computational Transition at the Uniqueness Threshold. In FOCS. 287–296.