Counting Database Repairs under Primary Keys Revisited* (DISCUSSION PAPER)

Marco Calautti, Marco Console, and Andreas Pieris

School of Informatics, University of Edinburgh
{mcalautt,mconsole,apieris}@inf.ed.ac.uk

1 Introduction

Consistent query answering (CQA) [2] aims to deliver meaningful answers when queries are evaluated over inconsistent databases. Such answers, called *consistent answers*, must be true in all repairs, which are consistent databases whose difference from the inconsistent one is somehow minimal. Consistent answers only allow us to conclude that a tuple is either entailed by all repairs, or is not entailed by some repair. But, as discussed in [4], the former is too strict, while the latter is not very useful in practice. Instead, we would like to know how often a tuple is consistent, i.e., its relative frequency defined as the ratio between the number of repairs entailing the tuple, and the total number of repairs.

The data complexity, i.e., when the query and the set of constraints are fixed, of the problem of computing the relative frequency of a tuple has been already studied in [6, 7] for conjunctive queries (CQs) and primary keys (i.e., when each relation has at most one key). We know that computing the total number of repairs is feasible in polynomial time. On the other hand, computing the number of repairs that entail the given tuple is #P-complete. Recall that #P is a hard counting complexity class, which, roughly speaking, collects the function problems that ask for the number of solutions to an NP problem. The above #P-hardness relies on polynomial-time Turing, a.k.a. Cook, reductions. However, as observed in the literature [5, 8], there are problems that are "hard-to-count-easy-to-decide" that cannot be complete (under reasonable assumptions) for #P under standard many-one logspace (or even polynomial-time) reductions. Note that the decision version of a counting problem asks whether the count is greater than zero. In our case, it asks whether a repair entailing a given tuple exists. As stated in [8], Cook reductions blur structural differences between counting problems and complexity classes. The reason is that #P is not closed under Cook reductions (under reasonable assumptions). Therefore, for such "hard-to-count-easy-to-decide" problems, a crucial question is whether

* This is a short version of the paper [3].

we can determine their exact complexity w.r.t. subclasses of #P to which they belong. Ideally, we would like to show that such a problem is complete, under logspace reductions, for a class $\mathcal{C} \subseteq$ #P that is closed under such reductions.

Our goal is to perform such a refined analysis for the data complexity of the problem of counting the number of repairs that entail a given tuple in the case of primary keys. We show that for arbitrary first-order queries this problem remains #P-complete. For existential positive queries (and CQs as well), we show that our problem *cannot* be complete (under reasonable assumptions) for #P, or even known classes inside #P. This brings us to our main question whether we can isolate a complexity class $\mathcal{C}$ inside #P that is closed under logspace reductions for which our problem (in the case of positive existential queries) is complete under such reductions. Let us stress that there is no guarantee that such a class $\mathcal{C}$ exists. This is because we focus on the data complexity of our problem. This means, by convention, that we deal with a family of problems and not a single problem; as usual, each query and set of primary keys gives rise to a new problem. Furthermore, establishing a completeness result for a certain complexity class $\mathcal{C}$ boils down to showing that (i) every problem in this family belongs to $\mathcal{C}$, and (ii) there exists one problem that is $\mathcal{C}$-complete. But, there is no guarantee that there exists a problem in this family such that all the other problems of the family are logspace reducible to it. This essentially tells us that the complexity class $\mathcal{C}$ inside #P that we are looking might not exist.

Although we do not give a definite answer to our main question, we make a significant step towards such an answer. We isolate a hierarchy of new complexity classes inside #P, called the Λ-hierarchy, which is of independent interest, and show that if the class $\mathcal{C}$ that we are looking for exists, then it is one of the levels of the Λ-hierarchy. Moreover, whether $\mathcal{C}$ exists boils down to the question whether the Λ-hierarchy collapses. We conjecture that this is not the case.

2 Preliminaries

Relational Databases. A *(relational) schema* $\mathbf{S}$ is a finite set of relation symbols (or predicates) with associated arity. A *database* D over a schema $\mathbf{S}$ is a set of facts of the form $R(\bar{t})$, where $R \in \mathbf{S}$, and $\bar{t} = t_1, \ldots, t_n$ is a tuple of terms that are drawn from a countably infinite set $\mathbf{C}$ of constants. We write $\mathsf{dom}(D)$ for the *active domain* of D, that is, the set of constants occurring in D.

Primary Key Constraints. A *key constraint* (or simply *key*) κ over a schema $\mathbf{S}$ is an expression of the form $\mathsf{key}(R) = \{1, \ldots, m\}$, where $R \in \mathbf{S}$ has arity n, and $m \leq n$. Such an expression is called an R-key. A database D satisfies κ if, for every two facts $R(\bar{t}), R(\bar{s}) \in D$, $\bar{t}[i] = \bar{s}[i]$ for each $i \in \{1, \ldots, m\}$ implies $\bar{t} = \bar{s}$. We say that D is *consistent* w.r.t. a set Σ of keys, written $D \models \Sigma$, if D satisfies each key in Σ; otherwise, it is *inconsistent*. A set of *primary keys* Σ is a set of keys such that, for each predicate R, Σ has at most one R-key. For $\alpha = R(c_1, \ldots, c_n)$, the *key value* of α w.r.t. Σ, denoted $\mathsf{key}_\Sigma(\alpha)$, is defined as $\langle R, \langle c_1, \ldots, c_m \rangle \rangle$, if $\mathsf{key}(R) = \{1, \ldots, m\} \in \Sigma$, and $\langle R, \langle c_1, \ldots, c_n \rangle \rangle$ otherwise. Let $\mathsf{block}_\Sigma(\alpha, D) = \{\beta \in D \mid \mathsf{key}_\Sigma(\beta) = \mathsf{key}_\Sigma(\alpha)\}$, and $\mathsf{block}_\Sigma(D) = \{\mathsf{block}_\Sigma(\alpha, D) \mid \alpha \in D\}$.

A *repair* of D w.r.t. Σ is a database obtained by choosing one fact from each $B \in \mathsf{block}_\Sigma(D)$. We denote the set of repairs of D w.r.t Σ as $\mathsf{rep}(D, \Sigma)$.

Queries. A *first-order query* $Q(\bar{x})$ over a schema $\mathbf{S}$ is an expression $\{\bar{x} \mid \varphi\}$, where φ is a first-order formula with free variables $\bar{x}$ that mentions only atoms over $\mathbf{S}$. We write $\mathbb{FO}$, $\exists\mathbb{FO}^+$ and $\mathbb{CQ}$, for the class of first-order, existential positive, and conjunctive queries, respectively. The answer to $Q(\bar{x})$ over a database D, denoted $Q(D)$, is the set of tuples $\{\bar{c} \in \mathsf{dom}(D)^{|\bar{x}|} \mid D \models \varphi(\bar{c})\}$.

Complexity Toolbox. We recall well-known counting complexity classes, i.e., classes of counting functions of the form $\{0,1\}^* \to \mathbb{N}$. FP contains the functions that are computable in polynomial-time. #P contains the functions that compute the number of accepting computations of an NP machine. SpanL contains the functions that compute the number of *distinct valid* outputs of an NL transducer. The output of a computation is called valid if the machine halts in an accepting state. We know that SpanL $\subseteq$ #P, and the inclusion is strict (under reasonable assumptions) [1]. Consider two counting functions $f, g : \{0,1\}^* \to \mathbb{N}$. We say that f is *Cook reducible* to g, denoted $f \leq_{\mathrm{T}}^{\mathrm{p}} g$, if there exists a polynomial-time deterministic machine M, with access to an oracle for g, such that, for every $x \in \{0,1\}^*$, $f(x) = M(x)$. Moreover, we say that f is *many-one logspace reducible* to g, denoted $f \leq_{\mathrm{m}}^{\log} g$, if there exists a function $h : \{0,1\}^* \to \{0,1\}^*$ that is computable in logarithmic-space such that, for every $x \in \{0,1\}^*$, $f(x) = g(h(x))$. As usual, a complexity class $\mathcal{C}$ is *closed* under $\leq_{\mathrm{T}}^{\mathrm{p}}$ (resp., $\leq_{\mathrm{m}}^{\log}$) if, for every two functions f, g, $f \leq_{\mathrm{T}}^{\mathrm{p}} g$ (resp., $f \leq_{\mathrm{m}}^{\log} g$) and $g \in \mathcal{C}$ implies $f \in \mathcal{C}$.

3 Problem Statement

The main concern of this work is the problem of counting the number of repairs that entail a given tuple defined as follows; $\mathbb{Q}$ denotes a class of queries:

PROBLEM :	$\#\mathsf{CQA}(\mathbb{Q})$
INPUT :	A database D, a set Σ of primary keys, a query $Q(\bar{x}) \in \mathbb{Q}$, and a tuple $\bar{t} \in \mathsf{dom}(D)^{\lvert\bar{x}\rvert}$.
OUTPUT :	$\lvert\{D' \in \mathsf{rep}(D, \Sigma) \mid \bar{t} \in Q(D')\}\rvert$.

As usual, we are interested in the *data complexity* of $\#\mathsf{CQA}(\mathbb{Q})$, where the set of constraints and the query are fixed, and only the database and the tuple are part of the input. To make this notion more precise, we need to define the following problem, where $Q(\bar{x}) \in \mathbb{Q}$ is a query and Σ is a set of primary keys:

PROBLEM :	$\#\mathsf{CQA}(Q(\bar{x}), \Sigma)$
INPUT :	A database D, and a tuple $\bar{t} \in \mathsf{dom}(D)^{\lvert\bar{x}\rvert}$.
OUTPUT :	$\lvert\{D' \in \mathsf{rep}(D, \Sigma) \mid \bar{t} \in Q(D')\}\rvert$.

We say that $\#\mathsf{CQA}(\mathbb{Q})$ is $\mathcal{R}$-complete for $\mathcal{C}$ in data complexity, where $\mathcal{R}$ is a class of reductions and $\mathcal{C}$ a complexity class, if (i) for each query $Q \in \mathbb{Q}$ and set

Σ of primary keys, $\#\mathsf{CQA}(Q, \Sigma)$ is in $\mathcal{C}$, and (ii) there exists $Q' \in \mathbb{Q}$ and set Σ' of primary keys such that $\#\mathsf{CQA}(Q', \Sigma')$ is $\mathcal{R}$-complete for $\mathcal{C}$.

The decision version of $\#\mathsf{CQA}(\mathbb{Q})$, denoted $\#\mathsf{CQA}_{>0}(\mathbb{Q})$, simply asks whether $|\{D' \in \mathsf{rep}(D, \Sigma) \mid \bar{t} \in Q(D')\}| > 0$. The data complexity of $\#\mathsf{CQA}_{>0}(\mathbb{Q})$ is defined in the same way as for $\#\mathsf{CQA}(\mathbb{Q})$. Henceforth, for clarity, we focus on Boolean queries, but all the results hold even if we consider non-Boolean queries.

4 Complexity of #CQA

The complexity of $\#\mathsf{CQA}$ has been already studied in [6, 7], and we know that $\#\mathsf{CQA}(\mathbb{CQ})$ is $\leq^{\mathrm{p}}_{\mathrm{T}}$-complete for #P.[1] The above result relies on Cook reductions. However, as discussed in Section 1, there are problems that are "hard-to-count-easy-to-decide", which cannot be complete (under reasonable assumptions) for #P under many-one logspace (or polynomial-time) reductions. Hence, the first step is to understand for which classes of queries $\mathbb{Q}$, $\#\mathsf{CQA}(\mathbb{Q})$ is an "easy-to-decide" problem. After a preliminary analysis, we have concluded that in the case of arbitrary first-order queries our problem is not only "hard-to-count", but also "hard-to-decide". In fact, we can show that $\#\mathsf{CQA}_{>0}(\mathbb{FO})$ is $\leq^{\log}_{\mathrm{m}}$-complete for NP, and $\#\mathsf{CQA}(\mathbb{FO})$ is $\leq^{\log}_{\mathrm{m}}$-complete for #P. This indicates that #P is the right complexity for our problem when we focus on first-order queries.

The situation for positive existential queries is more interesting. We can show that in this case we are dealing with a typical "hard-to-count-easy-to-decide" problem. In particular, we can show that $\#\mathsf{CQA}_{>0}(\exists\mathbb{F}\mathbb{O}^+)$ is tractable, which in turn implies that $\#\mathsf{CQA}(\exists\mathbb{F}\mathbb{O}^+)$ is not $\leq^{\log}_{\mathrm{m}}$-complete for #P (unless P = NP). This tells us that for pinpointing the complexity of $\#\mathsf{CQA}(\exists\mathbb{F}\mathbb{O}^+)$ we should look for a subclass of #P, which is closed under logspace reductions. The obvious candidate is SpanL. Indeed, we can show that $\#\mathsf{CQA}(\exists\mathbb{F}\mathbb{O}^+)$ is in SpanL. However, $\#\mathsf{CQA}(\exists\mathbb{F}\mathbb{O}^+)$ is not $\leq^{\log}_{\mathrm{m}}$-complete for SpanL (unless L = NL). Not much is known about subclasses of SpanL for which our problem can be complete under logspace reductions. This brings us to our main question:

Research Question: *Is there a class $\mathcal{C} \subseteq$ SpanL, which is closed under logspace reductions, such that $\#\mathsf{CQA}(\exists\mathbb{F}\mathbb{O}^+)$ is $\leq^{\log}_{\mathrm{m}}$-complete for $\mathcal{C}$?*

Let us stress that since we focus on the data complexity of $\#\mathsf{CQA}(\exists\mathbb{F}\mathbb{O}^+)$, as discussed in Section 1, the above question is not trivial in the sense that the existence of the class $\mathcal{C}$ is not guaranteed. Although we cannot give a definite answer to this question, we make a significant step towards such an answer. We isolate a hierarchy of new complexity classes inside #P, called the Λ-hierarchy, and show the following; we write $\Lambda[k]$ for the k-th level of the hierarchy:

Theorem 1. *The following are equivalent:*

[1] All the results for $\#\mathsf{CQA}$ and $\#\mathsf{CQA}_{>0}$ in the rest of the paper concern their data complexity. Thus, for brevity, we will sometimes avoid to explicitly mention the term data complexity.

1. *There exists a class* $\mathcal{C} \subseteq$ SpanL*, closed under logspace reductions, such that* $\#\mathsf{CQA}(\exists\mathbb{FO}^+)$ *is* $\leq_{\mathrm{m}}^{\log}$*-complete for* $\mathcal{C}$*.*
2. *There exists* $k \geq 0$ *such that* $\#\mathsf{CQA}(\exists\mathbb{FO}^+)$ *is* $\leq_{\mathrm{m}}^{\log}$*-complete for* $\Lambda[k]$*.*
3. *The* Λ*-hierarchy collapses.*

The above theorem essentially tells that either $\#\mathsf{CQA}(\exists\mathbb{FO}^+)$ is $\leq_{\mathrm{m}}^{\log}$-complete for $\Lambda[k]$, for some $k \geq 0$, or there is no complexity class $\mathcal{C} \subseteq$ SpanL, closed under many-one logspace reductions, such that $\#\mathsf{CQA}(\exists\mathbb{FO}^+)$ is $\leq_{\mathrm{m}}^{\log}$-complete for $\mathcal{C}$. Thus, the right complexity class that characterizes the data complexity of our problem, if it exists, is one of the levels of the Λ-hierarchy.

5 The Λ-hierarchy

We now proceed to introduce our hierarchy of complexity classes. The main technical challenge is to understand how to limit the computational power of NL transducers. This will lead to our new complexity classes inside SpanL that allow us to establish Theorem 1. To this end, we give a semi-formal introduction to this hierarchy, by presenting a couple of structural properties enjoyed by the functions therein. For further details, we refer the reader to [3].

5.1 The Guess-Check-Expand Paradigm

There is a generic paradigm, which we call *guess-check-expand*, that an NL transducer can follow in order to place problems inside SpanL, assuming that the following structural properties are fulfilled:

Small Certificates. A solution (in our case, a repair that entails the query) is witnessed via a *small certificate* – by small we mean of logarithmic size – that is verifiable in logspace.

For the next property, we first need to give some auxiliary terminology. Given a sequence of sets $S_1, \ldots, S_n$, an *ℓ-selector* of it, for $\ell \geq 0$, is a sequence of pairs $\sigma = (i_1, e_1), \ldots, (i_\ell, e_\ell)$, where $1 \leq i_1 < \ldots < i_\ell \leq n$, and $e_j \in S_{i_j}$ for each $j \in \{1, \ldots, \ell\}$. Intuitively, a pair (i, e) in σ tells us to "keep the element e from the set S_i". We can then define the *cartesian product of $S_1, \ldots, S_n$ w.r.t. σ*, denoted $[S_1, \ldots, S_n]^\sigma$, as $S_1^\sigma \times \cdots \times S_n^\sigma$, where $S_i^\sigma = \{e\}$ if σ contains a pair (i, e), otherwise, $S_i^\sigma = S_i$. In simple words, $[S_1, \ldots, S_n]^\sigma$ is the cartesian product of $S_1, \ldots, S_n$ with the difference that ℓ sets are first replaced by a singleton set as dictated by the ℓ-selector σ. We can now discuss the second structural property. For an input instance x, we write $\mathsf{sol}(x)$ for the set of solutions, and $\mathsf{cert}(x)$ for the set of certificates that witness a solution of $\mathsf{sol}(x)$.

Solutions via Certificate Expansion. For every input x, there is a sequence of logspace computable sets $S_1, \ldots, S_n$, called *solution domains*, such that

$$|\mathsf{sol}(x)| \;=\; \left| \bigcup_{c \in \mathsf{cert}(x)} [S_1, \ldots, S_n]^{\sigma_c} \right|,$$

where σ_c is an ℓ-selector for $S_1, \ldots, S_n$ determined by the small certificate c. In fact, there exists a bijection enc from $\mathsf{sol}(x)$ to $\bigcup_{c \in \mathsf{cert}(x)} [S_1, \ldots, S_n]^{\sigma_c}$, and $enc(s)$ should be understood as an encoding of the solution s.

A function that enjoys the above structural properties can be shown to be in SpanL via a simple guess-check-expand algorithm:

1. (**Guess**) Guess a candidate small certificate c.
2. (**Check**) If c is not a valid certificate, then reject.
3. (**Expand**) Expand c into an encoding of a solution witnessed by c using $S_1, \ldots, S_n$ as follows: for $i = 1$ to n, if the ℓ-selector σ_c determined by c mentions (i, e), then output e; otherwise, guess $e \in S_i$ and output e.

It is clear that, on an input x, the above transducer uses logspace in the size of x since the small certificates are logspace verifiable, and the solution domains are logspace computable. Moreover, the number of distinct valid outputs of the transducer coincides with $\left|\bigcup_{c \in \mathsf{cert}(x)} [S_1, \ldots, S_n]^{\sigma_c}\right|$, and thus, with $|\mathsf{sol}(x)|$. A transducer of the form above is called k-guess-check-expand, for some integer $k \geq 0$, if the ℓ-selectors are such that $\ell \leq k$.

$\#\mathsf{CQA}(Q, \Sigma)$ via a Guess-Check-Expand Algorithm. It is interesting to observe that the problem $\#\mathsf{CQA}(Q, \Sigma)$, for some UCQ Q and set Σ of primary keys, enjoys the two structural properties defined above; recall that an existential positive query can be rewritten as a UCQ. More precisely, on input D:

- A small certificate for a repair of $\mathsf{rep}(D, \Sigma)$ that entails Q is a pair (Q', h), where Q' is a disjunct of Q, and $h : \mathsf{var}(Q') \to \mathsf{dom}(D)$, such that $h(Q') \subseteq D$ and $h(Q') \models \Sigma$.
- The sequence of solution domains corresponds to the sequence $B_1, \ldots, B_n$ of the sets of $\mathsf{block}_\Sigma(D)$, according to some arbitrary, but fixed ordering. The ℓ-selector $\sigma_{(Q',h)}$ for $B_1, \ldots, B_n$ determined by a certificate (Q', h) mentions the pair $(i, R(\bar{t}))$, i.e., it keeps the fact $R(\bar{t})$ from B_i, iff $h(Q') \cap B_i = \{R(\bar{t})\}$ and Σ has an R-key. The latter implies that ℓ is bounded by the number of atoms with a predicate that has a key over all disjuncts of Q, which does not depend on the input database D. Clearly, the number of repairs of $\mathsf{rep}(D, \Sigma)$ that entail Q is the number

$$\left| \bigcup_{(Q',h) \in \mathsf{cert}(D)} [B_1, \ldots, B_n]^{\sigma_{(Q',h)}} \right|.$$

Therefore, the fact that $\#\mathsf{CQA}(Q, \Sigma)$ is in SpanL can be shown via a k-guess-check-expand algorithm, as the one presented below, where k is the number of atoms in Q with a predicate with a key in Σ:

1. (**Guess**) Guess a candidate small certificate (Q', h).
2. (**Check**) If $h(Q') \not\subseteq D$ or $h(Q') \not\models \Sigma$, then reject.
3. (**Expand**) Expand (Q', h) into an encoding of a solution witnessed by (Q', h) as follows: for $i = 1$ to n, if $h(Q') \cap B_i = \{R(\bar{t})\}$ and Σ has an R-key, then output $R(\bar{t})$; otherwise, guess a fact $\alpha \in B_i$ and output α.

5.2 Semi-formal Definition and Properties

The guess-check-expand paradigm suggests a natural way to define a hierarchy of classes inside SpanL. Intuitively, the k-th level of such hierarchy is defined by considering k-guess-check-expand transducers. This intuition is at the core of the Λ-hierarchy. In particular, for every integer $k \geq 0$, $\Lambda[k]$ is the class of functions $f : \{0,1\}^* \to \mathbb{N}$ such that, for every $x \in \{0,1\}^*$, $f(x)$ is the number of valid distinct outputs of a k-guess-check-expand transducer M_f, with input x. The Λ-hierarchy is defined as the infinite union $\Lambda = \bigcup_{k \geq 0} \Lambda[k]$.

We proceed to discuss some key properties concerning the Λ-hierarchy. The first one states that each level of the hierarchy is closed under logspace reductions, which is crucial for our complexity analysis.

Proposition 1. *For each $k \geq 0$, $\Lambda[k]$ is closed under logspace reductions.*

The next result confirms that Λ is a hierarchy of classes inside SpanL:

Proposition 2. $\Lambda[0] \subseteq \cdots \subseteq \Lambda \subseteq \text{SpanL}$, *and* $\Lambda = \text{SpanL}$ *implies* $\text{L} = \text{NL}$.

The next result shows that, apart from $\Lambda[0]$ and $\Lambda[1]$, the rest of the hierarchy consists of "hard" complexity classes:

Proposition 3. $\Lambda[1] \subseteq \text{FP}$, *and* $\Lambda[2] \subseteq \text{FP}$ *implies* $\text{P} = \text{NP}$.

A direct corollary of Proposition 3 is that $\Lambda[1] \subsetneq \Lambda[2]$ (unless P = NP). We do not know whether the inclusions beyond $\Lambda[2]$ are strict. This essentially asks whether there exists an integer $k \geq 2$ such that the Λ-hierarchy collapses to its k-th level, i.e., whether $\Lambda = \Lambda[k]$. This is a non-trivial open question.

6 Establishing our Main Result

Having the Λ-hierarchy in place, we are now ready to explain how Theorem 1 is shown. To this end, we concentrate on a parameterized version of $\#\mathsf{CQA}(\exists\mathbb{F}\mathbb{O}^+)$, where the so-called keywidth of the query w.r.t. the set of primary keys is bounded by an integer $k \geq 0$. Formally, the *keywidth function (kw)* maps pairs of the form (Q, Σ), where $Q \in \exists\mathbb{F}\mathbb{O}^+$ and Σ is a set of primary keys, to the naturals such that $kw(Q, \Sigma) = |\{R(\bar{t}) \mid R(\bar{t}) \text{ occurs in } Q, \text{ and } \Sigma \text{ has an } R\text{-key}\}|$. Then:

PROBLEM :	$\#\mathsf{CQA}_k^{kw}(\exists\mathbb{F}\mathbb{O}^+)$
INPUT :	A database D, a set Σ of primary keys, and a query $Q \in \exists\mathbb{F}\mathbb{O}^+$ such that $kw(Q, \Sigma) = k$.
OUTPUT :	$\lvert\{D' \in \mathsf{rep}(D, \Sigma) \mid D' \models Q\}\rvert$.

The data complexity of the above problem is defined as expected. Then:

Theorem 2. *For each $k \geq 0$, $\#\mathsf{CQA}_k^{kw}(\exists\mathbb{F}\mathbb{O}^+)$ is $\leq_{\text{m}}^{\log}$-complete for $\Lambda[k]$.*

By exploiting Theorem 2, we can now establish our main result (Theorem 1):

(1) ⇒ (3). By hypothesis, there is a class $\mathcal{C} \subseteq$ SpanL such that $\#\mathsf{CQA}(\exists\mathbb{FO}^+)$ is $\leq_{\mathrm{m}}^{\log}$-complete for $\mathcal{C}$. Thus, for every $Q \in \exists\mathbb{FO}^+$ and set Σ of primary keys, $\#\mathsf{CQA}(Q, \Sigma) \in \mathcal{C}$. Moreover, there exists $\hat{Q}$ and $\hat{\Sigma}$ such that $\#\mathsf{CQA}(\hat{Q}, \hat{\Sigma})$ is $\leq_{\mathrm{m}}^{\log}$-complete for $\mathcal{C}$, which means that, for every $f \in \mathcal{C}$, $f \leq_{\mathrm{m}}^{\log} \#\mathsf{CQA}(\hat{Q}, \hat{\Sigma})$. By Theorem 2, $\#\mathsf{CQA}(\hat{Q}, \hat{\Sigma}) \in \Lambda[kw(\hat{Q}, \hat{\Sigma})]$. Since, by Proposition 1, $\Lambda[kw(\hat{Q}, \hat{\Sigma})]$ is closed under logspace reductions, we conclude that $f \in \mathcal{C}$ implies $f \in \Lambda[kw(\hat{Q}, \hat{\Sigma})]$, or, in other words, $\mathcal{C} \subseteq \Lambda[kw(\hat{Q}, \hat{\Sigma})]$. Hence, for every $Q \in \exists\mathbb{FO}^+$ and set Σ of primary keys, $\#\mathsf{CQA}(Q, \Sigma) \in \Lambda[kw(\hat{Q}, \hat{\Sigma})]$. Since, for every $k \geq 0$, there exists Q' and Σ' such that $\#\mathsf{CQA}(Q', \Sigma')$ is $\leq_{\mathrm{m}}^{\log}$-complete for $\Lambda[k]$, then $\bigcup_{k\geq 0} \Lambda[k] = \Lambda[kw(\hat{Q}, \hat{\Sigma})]$. Hence, the Λ-hierarchy collapses to its $kw(\hat{Q}, \hat{\Sigma})$-th level.

(3) ⇒ (2). By hypothesis, the Λ-hierarchy collapses to its k-the level, for some $k \geq 0$, i.e., $\Lambda = \Lambda[k]$. We proceed to show that (i) for every $Q \in \exists\mathbb{FO}^+$ and set Σ of primary keys, $\#\mathsf{CQA}(Q, \Sigma) \in \Lambda[k]$, and (ii) there exists $\hat{Q}$ and $\hat{\Sigma}$ such that $\#\mathsf{CQA}(\hat{Q}, \hat{\Sigma})$ is $\leq_{\mathrm{m}}^{\log}$-complete for $\Lambda[k]$. For showing (i), fix some query $Q \in \exists\mathbb{FO}^+$ and set Σ of primary keys. By Theorem 2, $\#\mathsf{CQA}(Q, \Sigma) \in \Lambda[kw(Q, \Sigma)]$. Since $\Lambda[kw(Q, \Sigma)] \subseteq \Lambda = \Lambda[k]$, we immediately get that $\#\mathsf{CQA}(Q, \Sigma) \in \Lambda[k]$. Statement (ii) holds trivially since, by Theorem 2, there exists a $\leq_{\mathrm{m}}^{\log}$-complete function for every level of the Λ-hierarchy.

(2) ⇒ (1). Since, for every $k \geq 0$, $\Lambda[k]$ is closed under logspace reductions (Proposition 1), and $\Lambda[k] \subseteq$ SpanL (Proposition 2), the claim follows.

The Next Step. Providing a firm answer to our main question boils down to understanding whether the Λ-hierarchy collapses. Our conjecture is that it does not collapse. Proving or disproving this conjecture is one of our priorities, which will complete the picture concerning the data complexity of $\#\mathsf{CQA}(\exists\mathbb{FO}^+)$.

Acknowledgements. This work was supported by the EPSRC grants S003800 EQUID, M025268 VADA, and N023056 MAGIC.

References

1. Àlvarez, C., Jenner, B.: A very hard log-space counting class. Theor. Comput. Sci. **107**(1), 3–30 (1993)
2. Arenas, M., Bertossi, L.E., Chomicki, J.: Consistent query answers in inconsistent databases. In: PODS. pp. 68–79 (1999)
3. Calautti, M., Console, M., Pieris, A.: Counting database reparis under primary keys revisited. In: PODS (2019), to appear
4. Calautti, M., Libkin, L., Pieris, A.: An operational approach to consistent query answering. In: PODS. pp. 239–251 (2018)
5. Durand, A., Hermann, M., Kolaitis, P.G.: Subtractive reductions and complete problems for counting complexity classes. Theor. Comput. Sci. **340**(3), 496–513 (2005)
6. Maslowski, D., Wijsen, J.: A dichotomy in the complexity of counting database repairs. J. Comput. Syst. Sci. **79**(6), 958–983 (2013)
7. Maslowski, D., Wijsen, J.: Counting database repairs that satisfy conjunctive queries with self-joins. In: ICDT. pp. 155–164 (2014)
8. Pagourtzis, A., Zachos, S.: The complexity of counting functions with easy decision version. In: MFCS. pp. 741–752 (2006)