Combined Complexity of Repair Checking and Consistent Query Answering

Sebastian Arming, Reinhard Pichler, and Emanuel Sallinger

Vienna University of Technology
{arming,pichler,sallinger}@dbai.tuwien.ac.at

1 Introduction

Database management systems (DBMS) allow the definition of several forms of integrity constraints (ICs) to specify restrictions on the data to be stored. The DBMS provides checks that the stored data indeed satisfies the ICs. However, in modern applications where data is integrated from several sources, violations of the ICs may arise even if the data in each single source satisfies the ICs. Hence, the handling of *inconsistent* data (i.e., data violating the given ICs) has evolved as an active field of research, see e.g., [3, 5] for surveys. The foundations of this research were laid by Arenas et al. in [2], where the concepts of a *repair* and of *consistent answers* play key roles.

Given a set of ICs C and a (presumably inconsistent) database instance D, a *repair* of D w.r.t. C is a database instance I which satisfies C and which *differs minimally* from D. Difference and minimality can be defined in several ways. We follow the approach of [2] where repairs are obtained from the original database by the insertion and deletion of tuples and minimality means that the symmetric set difference $\Delta(D, I)$ is minimal w.r.t. subset inclusion. More formally, let $\Delta(D, I) = (D \setminus I) \cup (I \setminus D)$. Then I is a repair of D w.r.t. C if $I \models C$ and there does not exist an instance I' with $I' \models C$ and $\Delta(D, I') \subsetneq \Delta(D, I)$.

The idea of *consistent query answers* is that even from an inconsistent database instance D, we can derive consistent information, namely those answers to a query that one would obtain over every repair I of D. More precisely, the set of consistent answers to a query Q for a given database D and ICs C is defined as $\bigcap\{Q(I) \mid I \text{ is a repair of } D \text{ w.r.t. } C\}$.

The following decision problems are thus crucial to deal with inconsistent data:

Repair Checking (RC)
Instance: Databases D and I and a set of constraints C
Question: Is I a repair of D w.r.t. C?

Consistent Query Answering (CQA)
Instance: A database D, a set of constraints C, and a Boolean query Q
Question: Is Q true in all repairs of D w.r.t. C?

Apart from few exceptions (above all [1, 4]), most research on these decision problems has focused on the data complexity, i.e., the ICs C and the query Q are considered as fixed and only the database D is allowed to vary. The goal of our work is a thorough analysis of the *combined complexity* of these two problems.

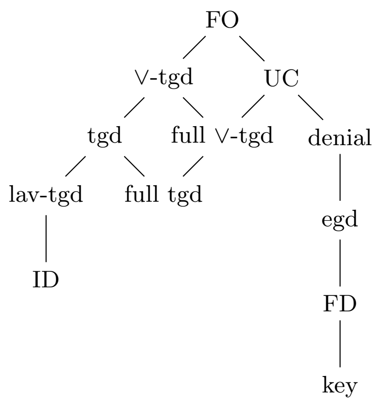
Hence, in addition to the database D, also the set of ICs C and (in case of the CQA-problem) the query Q are part of the input. As an important special case, we consider the complexity of these problems when the arity of the relation symbols is bounded by a constant.

As far as the queries Q are concerned, we concentrate on the fundamental class of Boolean conjunctive queries. It can be easily verified that all of our results carry over to arbitrary conjunctive queries (thus asking if a given tuple is contained in the answer to Q over every repair I) and unions of conjunctive queries. More powerful query languages are left for future work.

We consider a number of different languages from which the ICs C are taken. The languages considered here will range from First-Order logic as the most powerful one to inclusion dependencies and key dependencies which are among the least expressive ones. The contribution of this work is a fairly complete picture of the combined complexity of the RC and CQA problems.

2 Overview of Results

In this section, we first introduce the considered IC languages and after that give an overview of the complexity results for the RC and CQA problems. Figure 1 shows the hierarchy of the considered IC languages.



IC $\mathcal{L}$	RC	CQA	CQA b.a.
FO	PSPACE △▼	undec	undec
∨-tgd	Π_3P ▼	undec	undec
tgd	Π_3P ▲	undec	undec △
lav-tgd	DP ▲▼	NP ▽	NP
ID	in P ▼	NP	NP △
UC	Π_2P ▼	coNEXP-P$^{\text{NE}}$ ▼	Π_3P ▼
full ∨-tgd	Π_2P ▲	coNEXP-P$^{\text{NE}}$ ▲	Π_3P ▲
full tgd	DP ▲▼	EXP-coNEXP ▲▼	Π_2P ▲▼
denial	DP ▼	Π_2P ▽	Π_2P
egd	DP ▲	Π_2P	Π_2P
FD	in P ▼	Π_2P	Π_2P
key	in P	Π_2P	Π_2P △

Fig. 1. *Left:* Hierarchy of IC languages. *Right:* Combined complexity of RC, CQA and CQA with bounded arity. Black triangles indicate upper (▼) and lower (▲) bounds shown in this paper. White triangles indicate previously known bounds.

Besides *domain independent first order (FO)* sentences, all studied languages arise from restrictions on formulas of the form

$$\forall \boldsymbol{x}(\varphi(\boldsymbol{x}) \wedge \beta(\boldsymbol{x}) \rightarrow \bigvee_{i=1}^{n} \exists \boldsymbol{y_i} \psi_i(\boldsymbol{x}, \boldsymbol{y_i}))$$

where φ, ψ_i are conjunctions of database atoms and β a quantifier-free formula using only built-in (i.e. comparison) predicates. To ensure safety, we require that every variable in $\boldsymbol{x}$ must occur in some atom in φ. For simplicity, we assume that constraints do not contain constants (note however that our results still hold in the presence of constants).

We call such a constraint a *full* or a *universal constraint (UC)* if it contains no existential quantifiers. A *disjunctive tuple generating dependency ($\vee$-tgd)* has empty β, while an ordinary *tuple generating dependency (tgd)* additionally has $n = 1$. A *local-as-view (lav) tgd* is a tgd where φ is a single atom, and an *inclusion dependency (ID)* is a lav tgd where also ψ_1 is a single atom. A *denial constraint* is of the form $\forall \boldsymbol{x} \neg(\varphi(\boldsymbol{x}) \wedge \beta(\boldsymbol{x}))$ and can be thought of as a universal constraint with empty right hand side. An *equality generating dependency (egd)* is a denial constraint where β is a single inequality. A *functional dependency (FD)* is an egd where φ consists of two atoms over the same relation symbol. We recall that *key constraints* are a special case of FDs.

The table in Figure 1 shows the combined complexity of RC, CQA and CQA with bounded arity for each of the considered IC languages. Most results in the table are completeness results. In particular, if a single complexity class is listed in a cell, this denotes that the problem is complete for that class. Exceptions are that we do not further classify problems in P and undecidable problems. Furthermore, for the three cases where the exact complexity is not known, we list lower and upper bounds (e.g., EXP-coNEXP denotes an EXP lower bound and a coNEXP upper bound).

For showing the results claimed in Figure 1, it is not necessary to separately show membership and hardness for each single cell. The hierarchy shows the rich inclusion structure between the languages (e.g., every ID is a lav tgd). Similarly, CQA with bounded arity is a special case of CQA. Recall that lower bounds propagate from more restricted problems to more general problems and upper bounds propagate from more general to more restricted problems. Thus it suffices to show only certain membership and hardness results.

We use triangles in the table in Figure 1 to indicate the upper (▽/▼) and lower (△/▲) bounds that have to be shown in order to obtain all results given in the table. Black triangles indicate upper bounds (▼) and lower bounds (▲) shown in this paper. White triangles indicate upper bounds (▽) and lower bounds (△) given in previous work.

Theorem 1. *For the decision problems RC, CQA and CQA with bounded arity, the combined complexity is as depicted in the table in Figure 1.*

In Section 3, we will give the intuition of the results obtained in this paper (black triangles in Figure 1). In the remainder of this section, we summarize results given in or implicit in previous work (white triangles).

First, we consider the result known for RC. The PSPACE-hardness for FO follows immediately from the PSPACE-hardness of first-order model checking. Turning to CQA, [4] showed NP-completeness for IDs and mentioned Π_2P-completeness for key constraints. We note that the NP-membership already holds for lav tgds, and that the Π_2P-membership already holds for denial constraints. The final previously known result from Figure 1 is the undecidability of CQA for arbitrary tgds, shown in [8]. A close inspection of the hardness proofs for CQA shows that they already hold in case of bounded arity.

As a concluding remark, note that in Figure 1 we do not separately list RC with bounded arity. A quick inspection of our RC hardness proofs shows that the combined complexity of RC does not change when arity is bounded.

3 Intuition of Proofs

In this section, we discuss the results obtained in this work. For space reasons, we only give the intuition of the results. As membership results usually provide more insight into the sources of complexity, this section will mostly focus on membership results. More proof details can be found in the appendix.

3.1 Repair Checking

Repair checking has two fundamental sources of complexity, namely, checking that the supposed repair is consistent, and checking that it is indeed minimal. This immediately gives the following naive algorithm:

1. check consistency
2. try to guess a counter-example to minimality
   (a consistent database instance with smaller symmetric difference)

Since every database instance with a smaller symmetric difference has size at most polynomial in the size of the input, the guess is polynomial. Thus, if we know that the complexity of model checking of a constraint language is in M, then our naive algorithm yields an upper bound of $M \cap \mathsf{coNP}^M$. For $M = \mathsf{NP}$ (which is the case for lav tgds), the entire second step fits into coNP. Indeed, one can simultaneously guess a counter-example (a database instance) and a witness for its consistency. In this case, RC is in DP. For other classes M, the coNP^M factor dominates.

In many cases, one cannot do better than that. In particular, we use the upper bound given by the naive algorithm to show PSPACE-membership for FO, $\Pi_3\mathsf{P}$-membership for $\vee$-tgds, DP-membership for lav-tgds and $\Pi_2\mathsf{P}$-membership for UCs (four of the ▼ in the column RC of Figure 1). We will discuss the matching lower bounds later.

In some cases, one actually *can* do better than the naive algorithm. For full tgds and denial constraints, [7, Lemma 1] provides an NP test for checking minimality of a candidate repair. Since model checking for these classes is in coNP, this gives DP algorithms for RC.

For FDs, we exploit the fact that during the minimality check, it is sufficient to check the immediate supersets. This is the case since FDs can only be repaired by deletions and since they are monotone in the sense that supersets of inconsistent instances are always inconsistent. Since consistency can be checked in polynomial time, we thus have a polynomial time algorithm for RC.

For IDs, P-membership exploits the existence of a unique subset repair, i.e., a repair that can be obtained only by deletions. Such a subset repair can be computed in polynomial time in case of IDs. Using a construction similar to the one given in [8, Lemma 4.8], we can exploit subset repairs to devise a polynomial-time algorithm for the RC problem of IDs.

We now proceed to discussing our hardness results. We show $\Pi_3\mathsf{P}$-hardness for tgds as well as $\Pi_2\mathsf{P}$ hardness for full $\vee$-tgds by reduction from the corresponding QSAT problems. The DP-hardness results for lav tgds, full tgds, and egds are shown by reduction from 3-colorability and its complement.

An inspection of our proofs yields an interesting relationship between the roles of consistency and minimality checking, our two orthogonal sources of complexity. For tgds and full $\vee$ tgds, minimality checking dominates the complexity of RC. In particular, hardness holds even if the given instance is promised to be consistent. In contrast, for our DP results the role of consistency and minimality checking is distributed between the NP and the coNP parts of DP (i.e. if one check is NP, the other check is coNP). As a consequence, if the given instance is promised to be consistent, the complexity of RC for lav tgds drops to coNP while for gav tgds and egds it drops to NP. Finally, note that all results also hold for bounded arity.

3.2 Consistent Query Answering

The combined complexity of CQA with UCs was studied in [1], who showed $\mathsf{co2NEXP}$-membership and coNEXP hardness. We improve the upper bound to P^{NE} and show that already full $\vee$-tgds exhibit coNEXP-hardness.

We first illustrate the key ideas of the P^{NE}-membership proof for UCs. Let (D, Q, C) be an instance of CQA. The crucial observation is that it never makes sense to introduce fresh domain elements when repairing w.r.t. UCs. More precisely, let G be the set of all ground atoms over the active domain of D. Then every repair of D is a subset of G. We now give the following $\mathsf{NEXP}^{\mathsf{NP}}$ algorithm for the co-problem of CQA.

1. Guess $I \subseteq G$
2. Check that $I \not\models Q$ and $I \models C$
3. Call an oracle to check that there is no J such that $J \models C$ and that J has smaller symmetric difference to D than I

For verifying the complexity of our algorithm, observe that G has at most exponential size. Furthermore, note that checking whether a first-order formula φ is satisfied by a model M can be done in time $O\left(|\varphi|^2 \times |M|^{|\varphi|}\right)$. This model-checking algorithm can also be used inside the NP oracle by padding its input. Thus our algorithm is indeed in $\mathsf{NEXP}^{\mathsf{NP}}$. Since our algorithm uses only a single oracle call per computation path, it follows from the work of Hemaspaandra [6] that it belongs to the complexity class P^{NE}.

We now turn to the coNEXP-membership for full tgds. Note that the only purpose of the NP oracle in the above $\mathsf{NEXP}^{\mathsf{NP}}$ algorithm is to check the minimality of I. Recall that for full tgds, minimality can be checked in NP. By guessing the witness of this minimality check together with I, we can avoid the oracle call. This yields a coNEXP algorithm for CQA with full tgds.

The coNEXP hardness for full $\vee$-tgds and the EXP-hardness for full tgds can be shown by reductions from the evaluation problems of disjunctive Datalog and Datalog.

Turning to the final column of the table in Figure 1, note that if the arity of the relation symbols is bounded by a constant, the size G and therefore the size of any repair is polynomial in the size of the input. This observation, together with the complexity of the corresponding RC problems, immediately gives $\Pi_3\mathsf{P}$ membership for UCs and $\Pi_2\mathsf{P}$ membership for full tgds. Again, we provide matching lower bounds by reductions from the corresponding QSAT problems.

4 Conclusion

In this work, we have provided a fairly complete picture of the combined complexity of the RC- and CQA-problems. Only for the problem settings with EXP-hardness or above, a gap between the lower and upper bounds has remained. Future research should aim at closing this gap.

We have considered a wide range of constraint languages. However, in the literature, further classes of constraint languages can be found, such as binary constraints [5] or weakly acyclic tgds [8]. The exploration of the combined complexity of the RC- and CQA-problems for ICs from these classes has been left for future work. Yet more importantly, settings with combinations of various kinds of constraints (such as, e.g., inclusion dependencies with key dependencies) should be further explored, thus extending work that was already started in [4].

Finally, further problem variants deserve future investigation, such as adopting different notions of repairs, restricting the allowed repair actions, or considering different notions of minimality.

Acknowledgements. This work was supported by the Austrian Science Fund (FWF):P25207-N23 and by the Vienna Science and Technology Fund (WWTF) project ICT12-15.

References

1. Marcelo Arenas and Leopoldo Bertossi. On the decidability of consistent query answering. In *AMW*, 2010.
2. Marcelo Arenas, Leopoldo E. Bertossi, and Jan Chomicki. Consistent query answers in inconsistent databases. In *PODS*, pages 68–79. ACM Press, 1999.
3. Leopoldo Bertossi. Consistent query answering in databases. *ACM SIGMOD Record*, 35(2):68, 2006.
4. Andrea Calì, Domenico Lembo, and Riccardo Rosati. On the decidability and complexity of query answering over inconsistent and incomplete databases. In *PODS*, pages 260–271. ACM Press, 2003.
5. Jan Chomicki. Consistent Query Answering: Five Easy Pieces. In *ICDT*, pages 1–17, 2007.
6. Lane A. Hemachandra. The strong exponential hierarchy collapses. *J. Comput. Syst. Sci.*, 39(3):299–322, 1989.
7. Sławomir Staworko and Jan Chomicki. Consistent query answers in the presence of universal constraints. *Information Systems*, 35(1):1–22, 2010.
8. Balder ten Cate, Gaëlle Fontaine, and Phokion G. Kolaitis. On the data complexity of consistent query answering. In *ICDT*, pages 22–33. ACM Press, 2012.